Proceedings of First International Symposium on Sanskrit Computational Linguistics

Ed. Gérard Huet & Amba Kulkarni

INRIA, October 2007

Contents

Introduction

This volume contains the proceedings of the First International Symposium on Sanskrit Computational Linguistics, held at the Paris-Rocquencourt Research Center of INRIA form the 29th to the 31st of October 2007.

The grammatical tradition of Sanskrit and Computational Linguistics – both have a lot to offer to each other. This symposium provides a common platform to the traditional Sanskrit scholars and computational linguists to come together to share their knowledge, learn from and collaborate with the researchers from each other's disciplines. We hope this platform will result in a fruitful combination of two disciplines leading to new insights in the field of computational linguistics.

Computational Linguistics which started as an interdisciplinary branch a few decades ago, is today a full fledged branch of knowledge on its own. After 50 years of intensive research, impressive linguistic-based applications such as sophisticated search engines are in daily use. Although some technology is generic and can be parameterized properly for describing natural language structures in terms of formal linguistic models, in reality, such tools are available only for a select few languages. One crucial point is the development of linguistic resources, such as generative lexicons, treebanks of tagged reference corpuses, etc.

However, Sanskrit language offers a very interesting challenge for computational linguists:

- The Sanskrit language was studied to a high degree of formalization from high antiquity by genius linguists such as Pāṇini, and a continuous tradition of commenting and refining his work (Kātyāyana, Patañjali, Bhartṛhari, Nāgesha Bhaṭṭa, etc.), still very much alive, leaves hope for the emergence of new computational models of linguistic treatment, well tuned by definition to the Sanskrit language.

- The Sanskrit corpus contains a wealth of knowledge and wisdom (most of which is also available in electronic media) which has not been yet properly brought to light, despite centuries of both Western and Indian scholarship.

- Sanskrit benefits from meticulously checked data bases of verb forms, noun paradigm classes, and a host of other information necessary for building the computational tools for analysis and generation of Sanskrit texts.

- Yet the computational challenge of even simple tasks such as part-of-speech tagging has been a stumbling block hampering the use of computers for even simple philological tasks, such as finding all forms of a given root or morphological derivation in a text.

In view of recent progress reported for instance at the 13th World Sanskrit Conference in Edinburgh last July (`http://www.arts.ed.ac.uk/sanskrit/13thWSC/3participants.html`) and at the The First National Symposium on Modeling and Shallow Parsing of Indian Languages (MSPIL-06) in Mumbai last April (`http://www.cfilt.iitb.ac.in/~mspil-06/id25.htm`), it appeared to us that the time is ripe to coordinate our efforts at Computational Linguistics for Sanskrit. In quite a few sites worldwide, morpho-phonetic processing tools have been used to generate morphological banks, lemmatise forms, analyse sandhi, and even do some semantic processing. It would be nice if these various tools were interoperable, to the extent of being somehow composable. We would then get a leverage effect, by using their complementary capabilities. Efforts such as building Wordnet lexicons and tagged treebanks could be mutualized between the various teams. Mutual evaluation of the various tools would lead to their mutual improvement. This should not be done in an ad-hoc manner for Sanskrit, of course, it should follow proper standardisation of interchange formats, consistent with the international normalisation efforts along up-to-date technology (XML, Unicode, etc) for our work to be durable.

With this view in mind a core team of 6 members was formed and the team decided to hold a Symposium in order to benefit from the experience of other teams working in the area of Sanskrit Computational Linguistics worldwide.

This volume contains the papers selected for presentation at this First International Symposium on Sanskrit Computational Linguistics. The papers submitted were reviewed by at least three referees, and the reports were made available to the program committee members for open discussions if any. The reviews were well received and approved by the Program Committee. Ten papers were selected for presentation at the Symposium and one for discussion in the accompanying Workshop on standards for interoperable issues.

It is very encouraging that the selected papers cover a wide range of topics ranging from generating critical editions, tagging the existing corpus, developing various tools such as word analysers and generators, POS taggers and Parsers, applying modern computational tools such as Finite State technology to model the sandhi phenomenon in Sanskrit, studying contrastive linguistic phenomenon from different grammatical frameworks, and finally modeling Pāṇnini's monumental grammar, the Aṣṭādhyāyī.

We thank all the Program Committe members for their valuable support to make the symposium a success. We are grateful to Prof. Paul Kiparsky for accepting our invitation to deliver the invited conference.

Finally we thank all the researchers who responded to our call for papers and participants to this event without whose response the Symposium and Workshop would not have been a success.

We thank INRIA for the main financial support, and the other sponsors (Ecole des Hautes Études, UMR Mondes Iranien et Indien, University of Hyderabad Sanskrit Department, Rashtriya Sanskrit Vidyapeetha Tirupati and Rashtriya Sanskrit Sansthan) for additional invaluable support. Mrs Chantal Girodon from the Paris-Rocquencourt INRIA Research Center provided local organization, we thank her for insuring a smooth administrative operation.

Paris, October 2007,
Gérard Huet & Amba Kulkarni

Program Committee

SPONSORED BY

Figure 1: INRIA

Figure 2: Univ of Hyderabad

Figure 3: EPHE

Figure 4: EPHE

Figure 5: Rashtriya Sanskrit Vidyapeetha, Tirupati

Figure 6: Rashtriya Sanskrit Sansthan, Delhi

Conference Program

Monday October 29th 2007

Opening session (10 am - 10:45 am)

Invited lecture by Pr. Paul Kiparsky (11 am - 12 noon)

Session 1(2pm - 4 pm)

Tuesday October 30th 2007

Session 2(10 am - 12 noon)

Session 3(2 pm - 4 pm)

Workshop(4:15 pm - 5:45 pm)

Wednesday October 31st 2007

Session 4(10 am - 12 noon)

Session 5(2 pm - 4 pm)

Workshop(4:15 pm - 5:45 pm)

EXOCENTRIC (BAHUVRĪHI) COMPOUNDS IN CLASSICAL SANSKRIT

Brendan S. Gillon
McGill University

October 10, 2007

1. INTRODUCTION

Constituency grammars originated with Leonard Bloomfield (1933) and were developed during the nineteen forties and nineteen fifties by a number of American structuralist linguists, including Harris (1946), Wells (1947) and Hockett (1954) — to mention but a few. In the late nineteen fifties, Chomsky (1957) suggested that constituency grammars could be formalized as context free grammars. It is now clear that context free grammars fall short of properly formalizing the constituency grammars of the American Structuralists (Manaster-Ramer and Kac 1990). Indeed, in the nineteen sixties, Chomsky himself formalized a number of other important aspects of constituency grammars, introducing more complexity to the labels, both terminal and non-terminal, and permitting the use of null elements.

Though constituency grammars were initially conceived of as applying to phrases, work in the nineteen eighties (Di Sciullo and Williams 1987; Selkirk 1982; Williams 1981) showed that such rules could be used to profitably analyze compounds and derivational morphology of English. Gillon (1995) showed that the same analysis extended to the analysis of compounds and derivational morphology in Classical Sanskrit.

The aim of this paper is to look more carefully at the application of constituency rules to the analysis of exocentric (*bahuvrīhi*) compounds. In particular, I wish to show that the treatment of exocentric (*bahuvrīhi*) compounds in classical Sanskrit requires all of the enrichments of context free grammars which linguists think to be required. In particular, I shall show that exocentric compounds are nicely analyzed with the use of a phonetically null suffix. I shall also show that argument frames, a generalization of subcategorization frames, also play a key role in their analysis; these argument frames also play, as I shall show, a key role is providing a satisfactory analysis of so-called non-constituent compounds, a kind of compound of Classical Sanskrit, long recognized by the Indian grammatical tradition as problematic.

2. EXOCENTRIC (*BAHUVRĪHI*) COMPOUNDS

The classical Indian grammatical tradition identifies a number of different kinds of exocentric (*bahuvrīhi*) compounds. They include: privative exocentric (*nañ-bahuvrīhi*) compounds, comitative exocentric (*saha-bahuvrīhi* compounds, prepositional exocentric (*prādi-bahuvrīhi*) compounds, homo-denotative exocentric (*samānādhikaraṇa-bahuvrīhi*) compounds, and hetero-denotative exocentric (*vyadhikaraṇa-bahuvrīhi*) compounds. Our attention will be confined to homo-denotative ones (*samānādhikaraṇa-bahuvrīhi*).

Homo-denotative exocentric (*samānādhikaraṇa-bahuvrīhi*)[1] compounds are compounds whose canonical phrasal paraphrase is a relative clause in which the first constituent of the compound is predicated of the second, and they

[1]Homo-denotation (*sāmānādhikaraṇya*) is the counterpart in the Indian grammatical tradition of the Western technical notion of concord or agreement. As will be elaborated below, adjectives which modify nouns in Sanskrit agree with the nouns in case, number and gender. The concord is seen by the traditional Indian grammarian as accruing to the fact that a noun and an adjective modifying it have the same

thereby share the first, or nominative, case – and if the first constituent is an adjective, they agree in number and gender as well. Notice that the relationship of agreement in case, and where possible number and gender as well, is true of the canonical phrasal paraphrase not only of exocentric compounds, but also of descriptive compounds (*karmadhāraya*). Finally, homo-denotative exocentric compounds are so-called because they modify other constituents much in the way adjectives modify nouns. Putting these observations together, one arrives at the natural hypothesis that exocentric compounds are derived from descriptive compounds by zero affixation which converts a descriptive compound into an adjective.

(1)	Exocentric (*bahuvrīhi*) Compound:
Compound:	samacittaḥ (even-minded)
Analysis:	$((_A$ sama)≺$(_N$ cittaḥ$))_1$
	$((_N$ even)≺$(_N$ mind))
Paraphrase:	$[_{RC}$ $[_{VP}$ (asti)
	(is)
	$[_{AP1}$ samam] $[_{NP_1}$
	even
	cittam] yasya]
	mind whose
	whose mind is even

The evidence that homo-denotative exocentric (*samānādhikaraṇa-bahuvrīhi*) compounds are adjectives is that they have all the properties adjectives in Sanskrit have. First, adjectives in Sanskrit, like those in Latin, agree with the nouns they modify in case, number, and gender. Consider the adjective *tīkṣṇa* (*sharp*). If it modifies a noun in the nominative, singular, masculine, say *asiḥ* (*sword*), then it has the form *tīkṣṇaḥ*; and if it modifies a noun in the nominative, singular, feminine, say *churī* (*knife*), then it has the form *tīkṣṇā*; and if it modifies a noun in the nominative, singular, neuter, say *patram* (*blade*), then it has the form *tīkṣṇam*. Now consider the compound *dīrgha-kaṇṭha*. If it is to be construed with a masculine, singular noun in the nominative case, say *puruṣaḥ* (*man*), to yield the sense *long-necked man*, then the compound must have the nominative, masculine, singular form, namely, *dīrgha-kaṇṭhaḥ*. If it is to be construed with a feminine, nominative, singular noun, say *strī* (*woman*), to yield the sense *long-necked woman*, then the compound must have the feminine, nominative, singular form, *dīrgha-kaṇṭhā*. And finally, if it is to be construed with a neuter, nominative, singular noun, say *mitram* (*friend*), to yield the sense *long-necked friend*, then the compound must have the neuter, nominative, singular form, *dīrgha-kaṇṭham*.

Next, adjectives in Sanskrit can be turned into abstract nouns by the affixation of the suffix *-tva* (*-ness*): for example, the adjective *kṛśa* (*thin*) may be converted into the abstract noun, *kṛśa-tva* (*thin-ness*). Exocentric *bahuvrīhi* compounds are susceptible of the same conversion: for example, *dīrgha-kaṇṭha* (*long-neck-ed*; cf., *level-head-ed*) be turned into *dīrgha-kaṇṭha-tva* (*long-neck-ed-ness*; cf., *level-head-ed-ness*).

Moreover, just as an adjective such as *kṛśaḥ* (*thin*) can function, as its English translation can, as a common noun, meaning the same thing as its English nominal counterpart, *the thin*, so too should an exocentric (*bahuvrīhi*) compound be liable to function as a common noun. And this too is true, as observed by Speijer (1886, §222, fn. 1) and as exemplified by following compound and its commentarial gloss.

(2.1) NBT 48.4
(vyutpanna≺saṁketasya)

---

denotation (*samānādhikaraṇa*). A homo-denotative exocentric compound is one in which the two principal overt constituents denote the same thing, that is, they appear in the same case, and if the first principal overt constituent is an adjective, they agree in case, number, and gender.

(2.2) NBTP 49.1-2
vyutpannaḥ jñātaḥ saṁketaḥ yena saḥ
arisen known convention by whom he
One by whom the conventions of language are known
(*jñāta* (*known*) glosses *vyutpanna* (*arisen*).)

There is independent confirmation that homo-denotative exocentric (*samāna-adhikaraṇa-bahuvrīhi*) compounds are best treated as descriptive (*karmadhāraya*) compounds to which a phonetically null, possessive, adjectival suffix (symbolized hence forth with 'B') is affixed. Sanskrit has a phonetically overt, possessive, adjectival suffix -*ka* which is virtually synonymous with the phonetically null one just hypothesized. Though their distributions are somewhat different (A 5.4.151 ff.), nonetheless, they overlap to such an extent that commentators to a text in which an exocentric (*bahuvrīhi*) compound occurs frequently repeat the compound, adding the -*ka* suffix to signal the fact that the compound in question is to be construed, not as a descriptive (*karmadhāraya*) compound, but as an exocentric (*bahuvrīhi*) compound (Boose and Tubb 1981, ch. 5, sec. 15).

English too has homo-denotative exocentric compounds. By and large, they are marked by the adjectival, possessive suffix, -*ed*. These English compounds, exemplified by such compounds as *longlegged*, *literal-minded*, and *two-footed*, have a distribution narrower than that of its counterpart in Sanskrit — a fact which will be dilated on below.

Not every English homo-denotative exocentric compound has the -*ed* suffix. In particular, English homo-denotative exocentric compounds which serve as proper names or epithets seem to require a phonetically null counterpart to the -*ed* suffix.[2] Examples of proper names are particularly common in children's stories. For example, in the children's movie, *Land Before Time*, the two dinosaurs which are the main characters are named, *big foot* and *long neck*, instead of *big-footed* and *long-necked*. Examples of epithets are such compounds as *red-head*, *dim-wit*, *hard-back*, etc., to which there correspond *red-headed*, *dim-witted, hard-backed*, and so forth. (See Marchand 1969, ch. 2, sec. 18 for other examples.) Moreover, it seems that the -*ed* suffix and its phonetically null counterpart are in free variation in exocentric compounds which are initial constituents in larger compounds: *long-necked bottle plant* and *long neck bottle plant* both denote plants for bottles whose necks are long.

Another parallel between English and Sanskrit homo-denotative exocentric (*samāna-adhikaraṇa-bahuvrīhi*) compounds is the predication relation in the canonical paraphrase may be metaphorical, instead of literal. Thus, in the compounds *candra-mukha* (*moon-faced*), *sthūla-caraṇa* (*club-footed*), and *ayo-muṣṭi iron-fisted*, a face (*mukha*) is likened unto a moon (*candra*), a foot (*caraṇa*) unto a club (*sthūla*), and a fist (*muṣṭi*) unto iron (*ayas*).

As noted by Di Sciullo and Williams (1987, p. 30), in English, constituents outside of a compound cannot be construed with constituents subordinate within a compound. This generalization is undoubtedly true of English compound formation, as illustrated by the contrast in the interpretability of the expressions in (3).

(3.1) ((man≺eating)≺shark)
(3.2) *(eating≺shark) of men

Interestingly, this generalization was regarded as true of Sanskrit by Pāṇini. His treatment of compounds is to pair them with canonical phrasal paraphrases with which they share a common derivational ancestor; in addition to their semantic relation, they bear the syntactic relations of having the same heads and of having the same constituency. Hence, the constituency of compounds mirrors that of their canonical phrasal paraphrases. A condition on compound formation is that two elements cannot undergo compounding, the deletion of morphology from the subordinate element, unless the two elements form a constituent (A 2.1.4). A consequence of this is that inflected lexical items exterior to a compound are not construable with subordinate constituents within it. The applicability of this rule is illustrated both by Patañjali, in his Mahābhāṣya, or Great Commentary, on Pāṇini's

[2] It is interesting to note in this connection that the Sanskrit suffix -*ka*, used to mark phonetically a *bahuvrīhi* compound, is said by Pāṇini (A 5.4.155) to be prohibited from affixation to *bahuvrīhi* compounds which serve as names.

Aṣṭādhyāyī (at A 2.1.1), and by Bhartṛhari, in his work on the semantics of Sanskrit (VP 3.14.46), with the following example:

(4.1) [$_{NP_1}$ ((ṛddha ≺ rāja)≺puruṣaḥ)]
rich king man
servant of a rich king

(4.2) *[$_{NP_1}$ [$_{AP_6}$ ṛddhasya] (rāja≺puruṣaḥ)]
of rich king-man
servant of a rich king

Thus, the expression in (4.1) is acceptable, whereas the one in (4.2) is not, as signalled by the asterisk.

Though the generalization holds of English compounds, it does not of Sanskrit compounds. Counter-examples are furnished both by Patañjali (MBh on A 2.1.1) and by Bhartṛhari (VP 3.14.47):

(5.1) [$_{NP_6}$ [$_{NP_6}$ Devadattasya] guroḥ] kulam
of Devadatta of teacher family

(5.2) (Devadatta≺guru)≺kulam
Devadatta-teacher-family

(5.3) [$_{NP_6}$ Devadattasya] (guru≺kulam)
of Devadatta teacher-family
Devadatta's teacher's family

Indeed, compounds appearing in configurations such as that in (5.3) are given a special name by Sanskrit grammarians: they call them *asamartha* compounds (i.e., non-constituent compounds). Moreover, these compounds are well attested in the classical literature. A study of over three-hundred sentences, chosen essentially at random from the Sanskrit corpus, reveals thirteen clear cases of non-constituent (*asamartha*) compounds. (See Appendix I in Gillon 1993.) And a study of the first approximately five-hundred sentences of a single text reveals forty-three clear cases. (See Appendix II in Gillon 1993.)[3]

Thus, for example, in the best known play by the finest dramatist of Sanskrit literature, Kalidāsa's *Śakuntalā*, one finds precisely these configurations.

(6) Ś 3.9.16 (= SG 3.1.6)
[$_{NP_1}$ [$_{NP_3}$ [$_{NP_7}$ tasyām] (snigdha≺dṛṣṭyā)]
on her fixed-gaze
(sūcita≺abhilāṣaḥ)-B]
indicated-affection-ed
.. whose affection was indicated by his gaze being fixed on her

Here, the past passive participle, *sūcita* (*indicated*), which is a subordinate constituent within the exocentric (*bahuvrīhi*) compound *sūcita≺abhilāṣaḥ* (**indicated-affectioned*: *whose affection was indicated*), is construed with the third, or instrumental, case noun phrase *tasyām snigdha≺dṛṣṭyā* (*by his gaze being fixed on her*). Moreover, this noun phrase itself exhibits a non-constituent (*asamartha*) compound, for the past passive participle, *snigdha* (*fixed*) is found as a subordinate constituent in the compound *snigdha-dṛṣṭyā* (*by fixed-gaze*: *by his gaze being fixed*), yet it is construed with *tasyām* (*on her*), a seventh, or locative, case noun phrase, for which the verb *snih* (*to fix*) subcategorizes.

One person to attempt to meet the challenge presented by these compounds to Pāṇini's grammar of Sanskrit was Bhartṛhari, who suggested that non-constituent (*asamartha*) compounds are limited to cases where the subordinate

[3]To put these frequencies in perspective, I should point out that non-constituent (*asamartha*) compounds occurred more frequently in each corpus taken separately or jointly than either indirect questions or relative clauses.

constituent in the compound expresses a relation. Bhartṛhari's insight is a deep one. The remainder of this paper is devoted to showing how this insight might be captured within constituency grammars and then applied to not only non-constituent compounds but also to exocentric ones. The main concept is that of an argument frame, or an enriched subcategorization frame.

An argument frame has its classical quantificational logic. In elementary model theory, the set of predicates is partitioned into cells of the same degree, or adicity. The set of predicates are partitioned into the family of sets, one of which comprises all the one-place predicates, another all the two-place predicates, etc. This partitioning of the predicates has two effects. On the one hand, it helps to determine which string of symbols is a well-formed formula and which is not; on the other hand, it determines what kinds of values can be assigned to it. Thus, if one is told that P is a two place predicate and that a, b and c are terms, then one knows that Pab is a formula and that neither Pa no $Pcba$ is. At the same time, one knows that P is to be assigned a set of ordered pairs in the model.

A similar effect can be achieved with a slight enrichment of subcategorization frames. Subcategorization frames were introduced into constituency grammar by Chomsky (1965 ch. 2.3.4; 2.4). They greatly simplified the constituency rules by having individual lexical items specify their complements. Subcategorization frames effectively formalized and generalized the lexicographical practice of distinguishing between transitive and intransitive verbs. Because the subcategorization frame of a word is silent about those constituents which are not its complements, the subcategorization frame of a verb does not specify that it has an argument corresponding to the subject of the clause in which it might occur. Argument frames will have this specification. As a result, argument frames will fulfill the same functions in constituency grammar which predicate adicity fulfills in classical quantificational logic. It specifies the arguments associated with the word and it constrains the value assigned to it in a model to those relations with a corresponding arity, or degree of the relation. For example, a verb such as *to die*, which is intransitive, and thereby corresponding to a monadic predicate of classical quantification logic, takes only one argument and is assigned a unary relation (a subset of the model's domain), while a verb such as *to admire*, which is transitive, and thereby corresponding to a dyadic predicate, takes two arguments and is assigned a binary relation (a set of ordered pairs of members of the model's domain). Such a specification accounts for the contrasts in acceptability of the sentences given below.

(7.1) Bill died.
(7.2) *Bill died Fred.

(8.1) *Mary admires.
(8.2) Mary admires Bill.

Moreover, just as each predicate of a given adicity is interpreted by a relation of a corresponding arity, so each relational word is interpreted by a relation of a corresponding arity. It is crucial that this correspondance be properly established. To see why, consider this example from model theory. Let R be a binary predicate and let a and b be individual constants. Let M be a model whose domain is $\{1, 2, 3\}$ and whose interpretation function i assigns 1 to a, 2 to b and the set of ordered pairs $\{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 1\rangle\}$ to R. The clause of the truth definition of an atomic formula guarantees the following: Rab is true if and only if $\langle i(a), i(b)\rangle \in i(R)$. It is crucial that the order of appearance of the individual constants a and b in the formula Rab be correlated with the ordered pair $\langle i(a), i(b)\rangle$, not with the ordered pair $\langle i(b), i(a)\rangle$. As the reader can easily verify, the ordered pair $\langle i(a), i(b)\rangle$ is a member of $i(R)$, but not the ordered pair $\langle i(b), i(a)\rangle$.

The situation in natural language is, of course, much more complex that the situation in logical notation. I do not have the space to elaborate on all the detail. I shall, therefore, concentrate on just the essentials required for the task at hand. I shall signal the argument frame using the notation of ordered sets. Often, the arguments are restricted. This is well known in the case of verbs and nouns and adjectives derived from them. Indeed, these restrictions are at the heart of Pāṇini's grammar, the *Aṣṭādhyāyī*, where they are known as *kāraka*, or *factors*; and they have been widely exploited in contemporary linguistic theory, variously named *valences* or *thematic roles*.

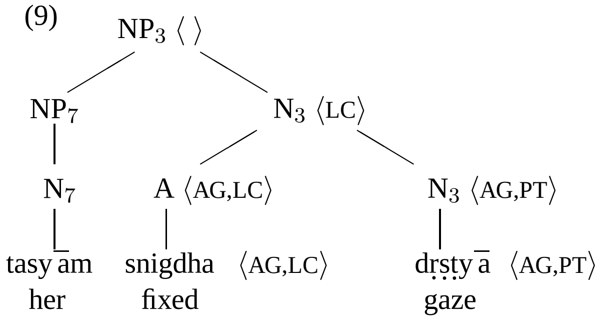
Typical valences include those recognized by Pāṇini: *agent*, *patient*, *beneficiary*, *source* and *location*.

It is important to stress that valences do not exhaust the kinds of restrictions which can be placed on arguments to lexical items. Underived relational nouns (for example, *friend*, *cousin*, *neighbor*, *colleague*) and underived relational adjectives (for example, *equivalent*, *opposite*, *proud*, *domestic* and *local*) have arguments, but their arguments are not plausibly said to be restricted by the usual valences associated with verbs.

Following Bhartṛhari's insight, I shall assume that non-constituent (*asamartha*) compounds appear when the subordinate constituent in the compound has an argument which is correlated with an inflected lexical item external to the compound. A survey of the cases mentioned above, as culled from the classical literature, shows that such is the case. Indeed, for the most part, the subordinate constituent is a deverbal noun or adjective, requiring an NP complement and often associating with it a particular valence.

Let us consider the case of a non-constituent compound *snigdha-dṛṣṭyā*, cited above. Recall that it is preceded by the pronoun *tasyām*, which is construed with the word *snigdha*, itself subordinate to *dṛṣṭyā*. The past passive participle *snigdha* has two arguments, one of which must appear in the seventh case.

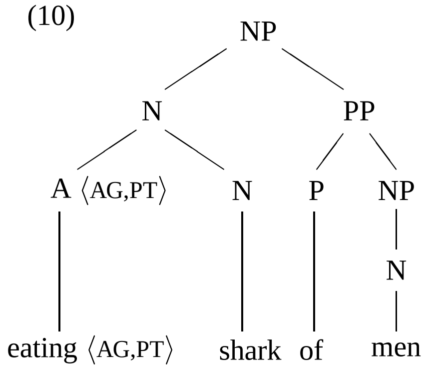
(9)



(where AG denotes *agent*, PT denotes *patient* and LC denotes *location*). The idea is that the argument frames are passed, as it were, up the tree. It is the *location* argument which is passed up to the top node of the tree for the compound.[4]

This contrasts with the situation in English. Sanskrit, as we just saw, permits unsaturated arguments associated with a non-head to be transmitted to the mother node, while English prohibits non-heads from having unsaturated arguments. Thus, for example, an expression such as (3.2) is prohibited in English. The reason is that, although one of the arguments associated with *eating*, namely the one whose valence is *agent* is saturated by the noun *shark*, the other argument associated with *eating*, namely the one whose valence is *patient* is not.[5]

(10)



[4]This compound is also an exocentric compound. This aspect of the compound is not being addressed h ere.

[5]Evidence that the argument with the valence of PATIENT is relevant comes from the acceptability of (3.1), namely *man-eating shark*.

This treatment of non-constituent (*asamartha*) compounds extends to exocentric (*bahuvrīhi*) compounds. Let us consider the following exocentric (*bahuvrīhi*) compounds in Sanskrit:

(11.1) SK 830
Compound: prāptātithiḥgrāmaḥ
Analysis: (prāpta≺atithiḥ-B) grāmaḥ
reached-guest-ed village
Paraphrase [$_{RC}$ atithayaḥ prāptāḥ yam]
guest reached which
saḥ prāptātithiḥgrāmaḥ
that village
the village which guests have reached

(11.2) SK 830
Compound: ūḍharathaḥ anaḍvān
Analysis: (ūḍha≺rathaḥ)-B anaḍvān
drawn-cart-ed bull
Paraphrase: [$_{RC}$ rathaḥ ūḍhaḥ yena]
cart bull by which
saḥ ūḍharathaḥ anaḍvān
that bull
the bull by which a cart is drawn

(11.3) SK 830
Compound: upahṛtapaśuḥ puruśaḥ
Analysis: (upahṛta≺paśuḥ)-B puruśaḥ
offered-cattle-ed man
Paraphrase: [$_{RC}$ paśuḥ upahṛtaḥ yasmai]
cattle offered to whom
saḥ upahṛta-paśuḥ puruśaḥ
that man
the man to whom cattle is offered

(11.4) SK 830
Compound: uddhṛtaudananā sthalī
Analysis: (uddhṛta≺odananā)-B sthalī
removed-rice vessel
Paraphrase: [$_{RC}$ odanaḥ uddhṛtaḥ yasyāḥ]
rice removed from which
sā uddhṛtaudananā sthalī
that vessel
the vessel from which rice has been removed

(11.5)	SK 830
Compound:	pītāmbaraḥpuruṣaḥ
Analysis:	(pīta-≺ambaraḥ)-B puruṣaḥ
	yellow-garment-ed man
Paraphrase:	[$_{RC}$ pītam ambaram yasya]
	yellow garment whose
	saḥ pītāmbaraḥpuruśaḥ
	that man
	the man whose garments are yellow

In the paraphrase and translation of Sanskrit exocentric (*bahuvrīhi*) compounds, the relative pronoun of the paraphrasing relative clause may be construed with either the subject (11.5) or the predicate (all other examples). Thus, the relative pronoun is construed with the predicate in such a way as to express the goal in the first example, the agent in the second, the beneficiary in the third, the source in the fourth, and the location in the last.

Indeed, as noted by Coulson (1976, p. 121), Sanskrit exocentric (*bahuvrīhi*) compounds are ambiguous between two readings: on one, the denotation of the lexical item modified by the exocentric compound is interpreted as the possessor of what is denoted by the final constituent of the compound; and on the other, it is interpreted as bearing a valence of any unsaturated argument associated with the initial constituent of the compound.

(12)	Coulson 1976, p. 121
Compound:	dṛṣṭakaṣṭā strī
Analysis:	(dṛṣṭa-kaṣṭā)-B strī
	witnessed-misfortune-ed woman
Reading 1:	a woman whose misfortune has been witnessed
	(i.e., a woman whose misfortune people have witnessed)
Reading 2:	a woman by whom misfortune has been witnessed
	(i.e., a woman who has witnessed misfortune)

Moreover, an exocentric compound has available a reading corresponding to each of the unsaturated arguments associated with its initial constituent.

(13)	Coulson 1976, p. 121
Compound:	dattādarā rajñī
Analysis:	(datta-ādarā)-B rajñī
	given-respect-ed queen
Reading 1:	a queen by whom respect is given
	(i.e, a respectful queen)
Reading 2:	a queen to whom respect is given
	(i.e., a respected queen)

Here emerges the difference between English and Sanskrit exocentric compounds alluded to above. Notice that, of the six examples, only the fifth allows an acceptable English calque: **reached-guested*, **drawn-carted*, **offered-cattled* and **removed-vesseled* but *yellow-garmented*. At the same time, while an English exocentric compound is paraphrasable with a relative clause, yet the relative pronoun of the paraphrase, "whose", is construed only with the subject of the relative clause, which corresponds to the final constituent of the compound paraphrased. Thus, *mean-spirited* is paraphrasable as *one whose spirit is mean*, *level-headed* as *one whose head is level*, and *long-legged* as *one whose legs are long*.

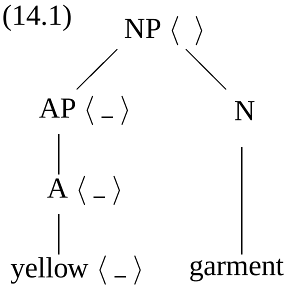
English and Sanskrit exocentric (*bahuvrīhi*) compounds differ as follows: the English adjectival suffix -*ed* does not permit the transmission of unsaturated arguments of an exocentric compound's initial constituent; whereas the

Sanskrit adjectival suffix B does permit the transmission of such arguments.

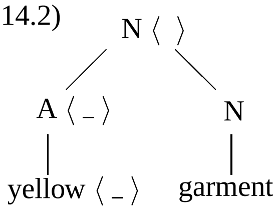
The foregoing differences between compounds in English and Sanskrit suggests the following hypothesis: the argument frame of initial constituents in lexical structure, in particular, in compound, percolate in Sanskrit but does not in English. This hypothesis accounts for two facts: first, that, in Sanskrit, unsaturated arguments associated with the initial constituent of an exocentric compound can be assigned to the lexical item the compound modifies, whereas in English they cannot be; second, that Sanskrit productively forms non-constituent (*asamartha*) compounds whereas English does not. Let us see how this account works.

Each adjective has at least one argument which is saturated either by the noun it modifies or by the subject noun phrase of which it is predicated. This is illustrated below, for modification both within phrasal structure and within compound structure.

(14.1)

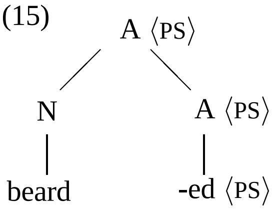


(14.2)



Now, both the *-ed* suffix in English and the -B suffix in Sanskrit create adjectives from nouns. This means that they create an argument. Associated with the resulting argument is the valence *possessor* (annotated PS). When the English suffix is applied to a simple noun like *beard*, one obtains the following:

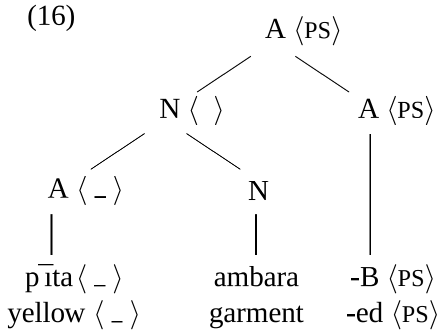
(15)



And when the resulting form modifies a word such as *man*, the resulting interpretation is *man who possesses a beard*. Combining what has been said so far, one obtains an analysis for both the Sanskrit compound in (11.5) and its English claque translation.

Moreover, the foregoing analysis shows precisely where Sanskrit and English differ. A morphologically complex English word accepts unsaturated arguments associated only with its head. Whereas, a morphologically complex Sanskrit word accepts the unsaturated arguments either of its head or of its head's sister. When an exocentric compound has no unsaturated argument other than the one associated with its possessive suffix, then its English and Sanskrit versions are equally acceptable.

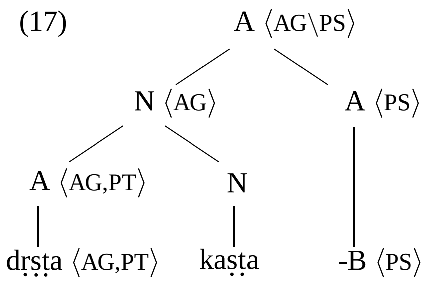
(16)



In this example, the argument associated with *yellow* (*pīta*) is saturated by *garment* (*ambara*), and so the complex word *yellow garment* (*pītāmbara*) has no unsaturated argument. The suffixation of *-ed* (-B) to *yellow garment* (*pītāmbara*) creates an unsaturated argument with an associated valence, namely that of possessor (PS).

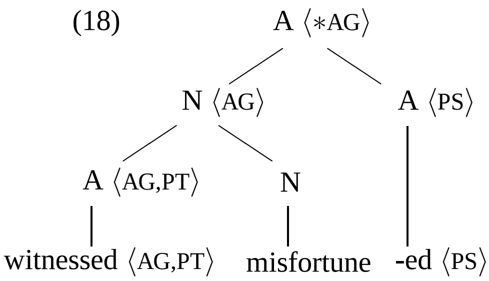
The situation is otherwise when the left-hand constituent of an exocentric compound has an unsaturated argument. Sanskrit permits unsaturated arguments associated with either a head and a non-head to be transmitted to the mother node; and, depending on which unsaturated argument is transmitted, the compound receives one or another interpretation. Thus, in the compound in (12) the unsaturated argument associated with the entire compound may have associated with it either the value AG or the value PS (annotated below as ⟨AG\PS⟩).

(17)



In constrast, English prohibits any unsaturated arguments from being associated with a non-head, with the consequence that the English counterparts to (9) are ungrammatical (annotated below as ⟨∗⟩).

(18)



3. CONCLUSION

Above, we examined two kinds of compounds in Classical Sanskrit, non-constituent (*asamartha*) compounds and exocentric (*bahuvrīhi*) compounds. The former compounds were considered problematic by the Indian grammati-

cal tradition for Pāṇini's grammar, the *Aṣṭādhyāyī*. An insight due to Bhartṛhari shows how they can be satisfactorily analyzed. This insight was recast using the notion of an argument frame, a generalization of subcategorization frame. A bonus of this solution is that it provides insight into well-know properties of the exocentric compounds of Classical Sanskrit, properties which exocentric compounds in English do not have.

4. WORKS CITED OR CONSULTED

Aklujkar, Ashok 1992 *Sanskrit: An Easy Introduction to an Enchanting Language*. Richmond, British Columbia (Canada): Svādhyāya Publications.

Apte, Vāman Shivarām 1885*The Student's Guide to Sanskrit Composition. A Treatise on Sanskrit Syntax for Use of Schools and Colleges*. Poona, India: Lokasamgraha Press, 24th edition (1960).

Apte, Vāman Shivarām 1890*The Practical Sanskrit-English Dictionary*. Poona, India: Prasad Prakashan, revised and enlarged edition (1957).

Bhaṭṭoji Dīkṣita *Siddhānta-Kaumudī*. Sanskrit Edition and English Translation: Vasu, Śrīśa Chandra (ed) (tr) 1906.

Bloomfield, Leonard 1933 *Language*. New York, New York: Holt.

Boose, Emery and Gary Tubb 1981 *Rough Draft of Portions of a Handbook Designed to Aid Students in the Use of Sanskrit Commentaries*. Unpublished mss., Harvard University.

Cardona, George 1988 *Pāṇini: His Work and Its Traditions. Background and Introduction*. New Delhi: Motilal Banarsidass.

Chomsky, Noam 1957 *Syntactic Structures*. The Hague, The Netherlands: Mouton and Company (Janua Linguarum: Series Minor n. 4).

Chomsky, Noam 1965 *Aspects of the Theory of Syntax*. Cambridge, Massachusetts: The MIT Press.

Coulson, Michael 1976 *Sanskrit: An Introduction to the Classical Language*. London, England: Hodder and Stoughton (Teach Yourself Books).

Deshpande, Madhav M 1985 *Ellipsis and Syntactic Overlapping: Current Issues in Pāṇinian Syntactic Theory*. Poona, India: Bhandarkar Oriental Research Institute (Post-graduate and Research Department Series no. 24).

Di Sciullo, Anna Maria and Edwin Williams 1987 *On the Definition of Word*. Cambridge, Massachusetts: The MIT Press.

Gillon, Brendan S. 1993 Bhartṛhari's solution to the problem of *asamartha* compounds. *Asiatische Studien/Études Asiatiques*: v. 47, n. 1, pp. 117–133.

Gillon, Brendan S. 1995 Autonomy of word formation: evidence from Classical Sanskrit. *Indian Linguistics*: v. 56, n. 1-4, pp. 15–52.

Godabole, N. B. (ed) 1933 *Śakuntalā*. Bombay: Nirṇaya Sāgara Press (revised by W. L. Paṇśīkar)

Harris, Zellig 1946 'From Morpheme to Utterance'. *Language*: v. 22, pp. 161-183.

Hockett, Charles F. 1954 'Two Models of Grammatical Description'. *Word*: v. 10, pp.210-231.

Iyer, K. A. Subramania (ed) 1973 *Vākyapadīya of Bhartṛhari with the Prakīrṇakaprakāśa of Helarāja*. Poona: Deccan College.

Iyyar, Seshādri (ed) 1896*Mālavikāgnimitra*. Poona, India: publisher unknown.

Kielhorn, F. (ed) 1880 *The Vyākaraṇa Mahābhāṣya of Patañjali*. Poona, India: Bhandarkar Oriental Research Institute (4^{th} edition revised by R. N. Dandekar 1985).

Kiparsky, Paul 1983 *Word Formation and the Lexicon*. In: Ingemann, Frances (ed) 1983 ??-??.

Malvania, Dalsukhabhai (ed) 1955 *Dūrveka Miśra's Dharmottarapradīpa*. Patna, India: Kashiprasad Jayaswal Research Institute (Tibetan Sanskrit Works Series: v. 2). 2^{nd} edition revised, 1971.

Manaster-Ramer, Alexis and Kac, Michael B. 1990 *Linguistics and Philosophy*: v. 13, pp. 325–362.

Marchand, Hans 1969 *The Categories and Types of Present-Day English Word-Formation. A Synchronic-Diachronic Approach*. Munich: C. H. Beck'sche Verlagsbuchhandlung. (reprint)

Monier-Williams, Monier 1899 *A Sanskrit-English Dictionary*. Oxford, England: Oxford University Press.

Selkirk, Elizabeth O. 1982. *The Syntax of Words*. Cambridge, Massachusetts: The MIT Press.

Speijer, J. S. 1886 *Sanskrit Syntax*. Leiden, The Netherlands: E. J. Brill.

Vasu, Śrīśa Chandra (ed) (tr) 1906 *The Siddhānta Kaumudī of Bhaṭṭoji Dikṣīta*. Allahabad, India: The Pāṇini Office. Reprint: Delhi, India: Motilal Banarsidass, 1962.

Wells, Rulon S. 1947 'Immediate Constituents' *Language*: v. 23, pp. 81-117.

Whitney, William Dwight 1881 *Sanskrit Grammar: Including both the Classical language, and the older Dialects, of Veda and Brahmana*. Cambridge, Massachusetts: Harvard University Press, 2nd edition (1889), 11th reprint (1967).

Williams, Edwin 1981 On the Notions "Lexically Related" and "Head of a Word". *Linguistic Inquiry* 12, 245-274.

FROM PĀṆINIAN SANDHI TO FINITE STATE CALCULUS

Malcolm D. Hyman
Max Planck Institute for the History of Science,
Berlin

ABSTRACT

The most authoritative description of the morphophonemic rules that apply at word boundaries (external sandhi) in Sanskrit is by the great grammarian Pāṇini (fl. 5th c. B. C. E.). These rules are stated formally in Pāṇini's grammar, the *Aṣṭādhyāyī* 'group of eight chapters'. The present paper summarizes Pāṇini's handling of sandhi, his notational conventions, and formal properties of his theory. An XML vocabulary for expressing Pāṇini's morphophonemic rules is then introduced, in which his rules for sandhi have been expressed. Although Pāṇini's notation potentially exceeds a finite state grammar in power, individual rules do not rewrite their own output, and thus they may be automatically translated into a rule cascade from which a finite state transducer can be compiled.

1. SANDHI IN SANSKRIT

Sanskrit possesses a set of morphophonemic rules (both obligatory and optional) that apply at morpheme and word boundaries (the latter are also termed *pada boundaries*). The former are called *internal sandhi* (< *saṃdhi* 'putting together'); the latter, *external sandhi*. This paper only considers external sandhi. Sandhi rules involve processes such as assimilation and vowel coalescence. Some examples of external sandhi are: *na asti* > *nāsti* 'is not', *tat ca* > *tac ca* 'and this', *etat hi* > *etad dhi* 'for this', *devas api* > *devo 'pi* 'also a god'.

This work has been supported by NSF grant IIS-0535207. Any opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views of the National Science Foundation. The paper has benefited from comments by Peter M. Scharf and by four anonymous referees.

The symbol ⟨'⟩ (*avagraha*) does not represent a phoneme but is an orthographic convention to indicate the prodelision of an initial *a*-.

2. SANDHI IN Pāṇini'S GRAMMAR

Pāṇini's *Aṣṭādhyāyī* is a complete grammar of Sanskrit, covering phonology, morphology, syntax, semantics, and even pragmatics. It contains about 4000 rules (termed *sūtra*, literally 'thread'), divided between eight chapters (termed *adhyāya*). Conciseness (*lāghava*) is a fundamental principle in Pāṇini's formulation of carefully interrelated rules (Smith, 1992). Rules are either *operational* (i. e. they specify a particular linguistic operation, or *kārya*) or *interpretive* (i. e. they define the scope of operational rules). Rules may be either obligatory or optional.

A brief review of some well-known aspects of Pāṇini's grammar is in order. The operational rules relevant to sandhi specify that a substituend (*sthānin*) is replaced by a substituens (*ādeśa*) in a given context (Cardona, 1965b, 308). Rules are written using metalinguistic case conventions, so that the substituend is marked as genitive, the substituens as nominative, the left context as ablative (*tasmāt*), and the right context as locative (*tasmin*). For instance:

8.4.62	*jhayo*	*ho*	*'nyatarasyām*
	jhaY-ABL	*h*-GEN	optionally

This rule specifies that (optionally) a homogenous sound replaces *h* when preceded by a sound termed *jhaY* — i. e. an oral stop (Sharma, 2003, 783–784). Pāṇini uses abbreviatory labels (termed *pratyāhāra*) to describe phonological classes. These labels are interpreted in the context of an ancillary text of the

The traditional classification of rules is more fine-grained and comprises *saṃjñā* (technical terms), *paribhāṣā* (interpretive rules), *vidhi* (operational rules), *niyama* (restriction rules), *pratiṣedha* (negation rules), *atideśa* (extension rules), *vibhāṣā* (optional rules), *nipātana* (ad hoc rules), *adhikāra* (heading rules) (Sharma, 1987, 89).

Aṣṭādhyāyī, the *Śivasūtras,* which enumerate a catalog of sounds (*varṇasamāmnāya*) in fourteen classes (Cardona, 1969, 6):

1. a i u Ṇ
2. ṛ ḷ K
3. e o Ṅ
4. ai au C
5. h y v r Ṭ
6. l Ṇ
7. ñ m ṅ ṇ n M
8. jh bh Ñ
9. gh ḍh dh Ṣ
10. j b g ḍ d Ś
11. kh ph ch ṭh th c ṭ t V
12. k p Y
13. ś ṣ s R
14. h L

The final items (indicated here by capital letters) are markers termed *it* 'indicatory sound' and are not considered to belong to the class. A *pratyāhāra* formed from a sound and an *it* denotes all sounds in the sequence beginning with the specified sound and ending with the last sound before the *it.* Thus *jhaY* denotes the class of all sounds from *jh* through *p* (before the *it Y*): *jh, bh, gh, ḍh, dh, j, b, g, ḍ, d, kh, ph, ch, ṭh, th, c ṭ, t, k, p* (i. e. all oral stops). Sūtra 8.4.62, as printed above, is by itself both elliptic and uninterpretable. Ellipses in sūtras are completed by supplying elements that occur in earlier sūtras; the device by which omitted elements can be inferred from preceding sūtras is termed *anuvṛtti* 'recurrence' (Sharma, 1987, 60). It will be noticed that no substituens is specified in 8.4.62; the substituens *savarnaḥ* 'homogenous sound-NOM' (Cardona, 1965a) is supplied by *anuvṛtti* from sūtra 8.4.58 *anusvārasya yayi parasavarṇaḥ.* Still, the device of *anuvṛtti* is insufficient to specify the exact sound that must be introduced as a substituens. It is here that interpretive rules play a role. Sūtra 8.4.62 must be interpreted in the light of the *paribhāṣā* 'interpretive rule' 1.1.50 *sthāne 'ntaratamaḥ*, which specifies that a substituens must be maximally similar (sc. in articulatory place and manner) to the substituend (Sharma, 1987, 126). Thus the substituens in 8.4.62 will always be aspirated, since *h* (the substituend) is aspirated: e. g. *vāg hasati* (< *vāk hasati* 'a voice laughs' by 8.2.39) > *vāgghasati.*

A second example further illustrates the principles already discussed:

8.4.63	*śaś*	*cho*	*'ṭi*
	ś-GEN	*ch*-NOM	*aṬ*-LOC

Here *jhayaḥ* '*jhaY*-ABL' and *anyatarasyām* 'optionally' are supplied by *anuvṛtti* from the preceding sūtra (8.4.62). The rule specifies that (optionally) *ś* is replaced by *ch* when it is preceded by an oral stop (*jhaY*) and followed by a vowel or semivowel (*aṬ*).

Rules specific to external sandhi are found in the third quarter (*pāda*) of the eighth *adhyāya.* A number of rules are common to both internal and external sandhi, and rules relevant for external sandhi are also found in the first *pāda* of the sixth *adhyāya* and the fourth *pāda* of the eighth *adhyāya.*

3. AN XML ENCODING FOR PāṇiniAN RULES

An encoding based on Extensible Markup Language (XML) has been chosen for expressing Pāṇinian rules in machine-readable form. The XML vocabulary contains an element `<rule>`, with required attributes `source` (the substituend) and `target` (the substituens) and optional attributes `lcontext` (the left context) and `rcontext` (the right context). The values of `source`, `lcontext`, and `rcontext` are specified as Perl-compatible regular expressions (PCREs) (Wall et al., 2000). Sanskrit sounds are indicated in an encoding known as SLP1 (Sanskrit Library Phonological 1) (Scharf and Hyman, 2007). An encoding such as Unicode is not used, since Unicode represents *written characters* rather than *speech sounds* (Unicode Consortium, 2006). The SLP1 encoding facilitates linguistic processing by representing each Sanskrit sound with a single symbol (see fig. 1). The use of Unicode would be undesirable here, since (1) there would not be a one-to-one correspondence

The vowel *a* added after a consonant makes the *pratyāhāra* pronounceable.

Note that Sanskrit *h* represents a voiced glottal fricative [ɦ].

So-called "regular expressions" in programming languages such as Perl include extended features such as pattern memory that exceed the power of regular languages (see §4); for example, it is possible to write a regular expression that matches the $\alpha\alpha$ language, that is, the language of all reduplicated strings. Thus PCREs are *not*, in the formal sense, regular expressions at all.

Figure 1 is a simplified overview of SLP1. It does not show symbols for accents, certain nasalized sounds, or sounds peculiar to the Vedic language.

between character and sound, and (2) Sanskrit is commonly written in a number of different scripts (Devanāgarī, Tamil, Romanization, etc.).

The following is the XML representation of sūtra 8.3.23 *mo 'nusvāraḥ*, which specifies that a *pada*-final *m* is replaced by the nasal sound *anusvāra* (*ṃ*) when followed by a consonant (Sharma, 2003, 628):

```
<rule source="m" target="M"
      rcontext="[@(wb)][@(hal)]"
      ref="A.8.3.23"/>
```

This rule employs a syntactic extension used for *macros*. The expression `@(name)` is replaced by the value of a defined macro `name`. Macros are defined with an XML element `<macro>` and may be defined recursively. Macro expansion is performed immediately after parsing the XML file, before any further processing. Here the `rcontext` attribute references two macros: `@(wb)` is expanded to characters that indicate a word boundary, and `@(hal)` is expanded to the characters representing the sounds of the *pratyāhāra haL* (i. e. all consonants). Macros are a syntactic convenience that allows rules to be easier to read (and closer to Pāṇini's original formulation); one might equally spell out in full all characters representing sounds in a phonological class. The square brackets belong to the PCRE syntax and indicate that any character contained between them should be matched; that is, `[abc]` matches an `a`, `b`, or `c`.

In the `target` two additional syntactic facilities allow for rules to be expressed in a way that is close to Pāṇini's formulation. These facilities are termed *mappings* and *functions*. The following rule illustrates a mapping:

```
<rule source="h"
      target="%(voicedaspirate($1))"
      lcontext="([@(Jay)])[@(wb)]"
      optional="yes"
      ref="A.8.4.62"/>
```

This sūtra has been discussed earlier. The left context matches a *jhaY* followed by a word boundary. The parentheses (part of PCRE syntax) specify that the contents (the *jhaY*) be stored in pattern memory. Strings stored in pattern memory are available for subsequent reference: the pattern matched by the first parenthesized group may be recalled with `$1`; the second, with `$2`, etc. (only nine pattern memory variables are available). The pattern memory variable `$1` is referenced in the `target` attribute. The rule specifies that an *h*, when preceded

A a **a**	A;a ā **A**	I i **i**	Rī **I**	o u **u**	'o ū **U**
'x ṛ **f**	'X ṝ **F**	'w ḷ **x**	'W ḹ **X**	*	
O; e **e**	Oe; ai **E**	A;ea o **o**	A;Ea au **O**		
ḳ k **k**	K,a kh **K**	g,a g **g**	;G,a gh **G**	.z ṅ **N**	
..c,a c **c**	Ç ch **C**	.j,a j **j**	J,a jh **J**	V,a ñ **Y**	
,f ṭ **w**	F ṭh **W**	.q ḍ **q**	Q ḍh **Q**	:N,a ṇ **R**	
t,a t **t**	T,a th **T**	ḍ d **d**	;D,a dh **D**	n,a n **n**	
:p,a p **p**	:P ph **P**	b,a b **b**	B,a bh **B**	m,a m **m**	
y,a y **y**	.=,r **r**	ḷ l **l**	v,a v **v**		
Z,a ś **S**	:S,a ṣ **z**	.s,a s **s**	ḥ h **h**		

* anusvāra = **M**; visarga = **H**

Figure 1: A partial overview of the SLP1 encoding

by a *pada*-final *jhaY*, is replaced by the result of performing the `voicedaspirate` mapping on the matched *jhaY*. The mapping syntax takes the form `%(name(input))`, where `name` is the name of a mapping, and `input` is the input symbol for the mapping. A mapping is defined with a `<mapping>` element, which has as children one or more `<map>` elements. The mapping `voicedaspirate` is defined thus:

```
<mapping name="voicedaspirate">
  <map from="@(jaS)" to="@(Jaz)"/>
</mapping>
```

This mapping translates the voiced oral stops denoted by the *pratyāhāra jaŚ* (*j, b, g, ḍ, d*) to the equivalent aspirated voiced oral stops denoted by the *pratyāhāra jhaṢ* (*jh, bh, gh, ḍh, dh*). In the case that the input symbol to a mapping is not contained in `from`, the mapping is equivalent to the identity function.

Sūtra 6.1.87 *ād guṇaḥ* illustrates the use of a function:

```
<rule source="[@(a)][@(wb)]([@(ik)])"
      target="!(gunate($1))"
      ref="A.6.1.87"/>
```

The `source` matches either short *a* or long *ā* (by the definition of the macro `a`), a word boundary, and then the *pratyāhāra iK* (a simple vowel other than *a* or *ā*). The macro `ik` is defined:

```
<macro name="ik"
       value="@(i)@(u)@(f)@(x)"
       ref="A.1.1.71"/>
```

and depends on the macro definitions:

```
<macro name="i" value="iI"
       ref="A.1.1.69"/>
<macro name="u" value="uU"
       ref="A.1.1.69"/>
<macro name="f" value="fF"
       ref="A.1.1.69"/>
<macro name="x" value="xX"
       ref="A.1.1.69"/>
```

The *iK* is stored in pattern memory, and the substituend (*a-varṇa*) is replaced by the output of calling the function `gunate` on the stored *iK*. A function is defined with an element `<function>`, which has as children one or more `<rule>` elements. These are context-free rules that typically make use of mappings in the `target`. Thus `gunate` is defined:

```
<function name="gunate">
  <rule source="[@(a)@(i)@(u)]"
        target="%(guna($1))"/>
  <rule source="[@(f)@(x)]"
        target="%(guna($1))
                %(semivowel($1))"/>
</function>
```

Two mappings are invoked here:

```
<mapping name="guna"
         ref="A.1.1.2">
  <map from="@(a)" to="a"/>
  <map from="@(i)" to="e"/>
  <map from="@(u)" to="o"/>
  <map from="@(f)" to="a"/>
  <map from="@(x)" to="a"/>
</mapping>

<mapping name="semivowel"
         ref="A.6.1.77">
  <map from="@(i)" to="y"/>
  <map from="@(u)" to="v"/>
  <map from="@(f)" to="r"/>
  <map from="@(x)" to="l"/>
</mapping>
```

The mapping `guna` maps simple vowels such as *a-varṇa* (which includes short *a* and long *ā*) to their *guṇa* equivalent. The term *guṇa* is defined by the technical rule (*saṃjñā*) (Sharma, 1987, 102) 1.1.2 *adeṅ guṇaḥ* '*a* and *eṄ* [are] *guṇa*' (the *pratyāhāra* $e\dot{N} = \{e, o\}$). The mapping `semivowel` maps those vowels that possess homorganic semivowels to the corresponding semivowels. The function `gunate`, if the input is *a-*, *i-*, or *u-varṇa,* outputs the corresponding *guṇa* vowel; if the input is *ṛ-* or *ḷ-varṇa,* it outputs the corresponding *guṇa* vowel (in this case, *a*) concatenated with the homorganic semivowel (either *r* or *l*). The domain of `gunate` is {*a, ā, i, ī, u, ū, ṛ, ṝ, ḷ, ḹ*} and its range is {*a, e, o, ar, al*}.

4. REGULAR LANGUAGES AND REGULAR RELATIONS

First, it is necessary to define a *regular language.* Let Σ denote a finite alphabet and Σ^ϵ denote $\Sigma \cup \{\epsilon\}$ (where ϵ is the empty string). $\{e\}$ is a regular language where $e \in \Sigma^\epsilon$, and the empty language $\emptyset$ is a regular language. Given that L_1, L_2, and L are regular languages, additional regular languages may be defined by three operations (under which they are closed): concatenation ($L_1 \cdot L_2 = \{xy|x \in L_1, y \in L_2$), union ($L_1 \cup L_2$), and Kleene closure ($L^* = \cup_{i=0}^{\infty} L^i$) (Kaplan and Kay, 1994, 338).

Regular relations are defined in the same fashion. An n-relation is a set whose members are ordered n-tuples (Beesley and Karttunen, 2003, 20). Then $\{e\}$ is a regular n-relation where $e \in \Sigma^\epsilon \times \ldots \times \Sigma^\epsilon$ ($\emptyset$ is also a regular n-relation). Given that R_1, R_2, and R are regular n-relations, additional regular n-relations may be defined by three operations (under which they are closed): n-way concatenation ($R_1 \cdot R_2 = \{xy|x \in R_1, y \in R_2$), union ($R_1 \cup R_2$), and n-way Kleene closure ($R^* = \cup_{i=0}^{\infty} R^i$).

5. FINITE STATE AUTOMATA AND REGULAR GRAMMARS

A *finite state automaton* is a mathematical model that corresponds to a regular language or regular relation (Beesley and Karttunen, 2003, 44). A simple finite state automaton corresponds to a regular language and is a quintuple $\langle S, \Sigma, \delta, s_0, F \rangle$, where S is a finite set of states, Σ the alphabet of the automaton, δ is a transition function that maps $S \times \Sigma^\epsilon$ to 2^S, $s_0 \in S$ is a single initial state, and $F \subseteq S$ is a set of final states (Aho et al., 1988, 114). A finite state automaton that corresponds to a regular relation is termed a *finite state transducer* (FST) and can be defined as a quintuple $\langle S, \Sigma \times \ldots \times \Sigma, \delta, s_0, F \rangle$, with δ being a transition function that maps $S \times \Sigma^\epsilon \times \ldots \times \Sigma^\epsilon$ to 2^S (Kaplan and Kay, 1994, 340).

A regular (or finite) grammar describes a regular language and is equivalent to a finite state automaton. In a regular grammar, all production rules have a single non-terminal on the left-hand side, and either a sin-

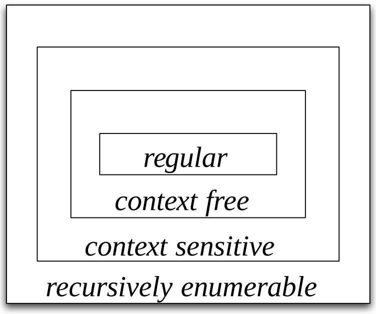


Figure 2: The Chomsky hierarchy of languages (after Prusinkiewicz and Lindenmayer (1990, 3))

gle terminal or a combination of a single non-terminal and a single terminal on the right-hand side. That is, all rules are of the form $A \rightarrow a$, $A \rightarrow aB$ (for a right regular grammar), or $A \rightarrow Ba$ (for a left regular grammar), where A and B are single non-terminals, and a is a single terminal (or ϵ). A regular grammar is the least powerful type of grammar in the Chomsky hierarchy (see fig. 2) (Chomsky, 1956). A context free grammar describes a context free language, a context sensitive grammar describes a context sensitive language, and an unrestricted grammar describes a recursively enumerable language. The hierarchy is characterized by proper inclusion, so that every regular language is context free, every context free language is context sensitive, etc. (but not every context free language is regular, etc.). A regular grammar cannot describe a context free language such as $\{a^n b^n | 1 \leq n\}$, which consists of the strings $\{ab, aabb, aaabbb, aaaabbbb \dots\}$ (Kaplan and Kay, 1994, 346).

Since Pāṇinian external sandhi can be modeled using finite state grammar, it is highly desirable to provide a finite state implementation, which is computationally efficient. Finite state machines are closed under composition, and thus sandhi operations may be composed with other finite state operations to yield a single network.

6. FROM REWRITE RULES TO REGULAR GRAMMARS

A string rewriting system is a system that can transform a given string by means of rewrite rules. A rewrite rule specifies that a substring $x_1 \dots x_n$ is replaced by a substring $y_1 \dots y_m$: $x_1 \dots x_n \rightarrow y_1 \dots y_m$, where $x_i, y_i \in \Sigma$ (and Σ is a finite alphabet). Rewrite rules used in phonology have the general form

$$\phi \rightarrow \psi / \lambda ___ \rho$$

Such a rule specifies that ϕ is replaced by ψ when it is preceded by λ and followed by ρ. Traditionally, the phonological component of a natural-language grammar has been conceived of as an ordered series of rewrite rules (sometimes termed a *cascade*) $W_1, \dots, W_n$ (Chomsky and Halle, 1968, 20). Most phonological rules, however, can be expressed as regular relations (Beesley and Karttunen, 2003, 33). But if a rewrite rule is allowed to rewrite a substring introduced by an earlier application of the same rule, the rewrite rule exceeds the power of regular relations (Kaplan and Kay, 1994, 346). In practice, few such rules are posited by phonologists. Although certain marginal morphophonological phenomena (such as arbitrary center embedding and unlimited reduplication) exceed finite state power, the vast majority of (morpho)phonological processes may be expressed by regular relations (Beesley and Karttunen, 2003, 419).

Since phonological rewrite rules can normally (with the provisos discussed above) be reexpressed as regular relations, they may be modeled as finite state transducers (FSTs). FSTs are closed under composition; thus if T_1 and T_2 are FSTs, application of the composed transducer $T_1 \circ T_2$ to a string S is equivalent to applying T_1 to S and applying T_2 to the output of T_1:

$$T_1 \circ T_2(S) = T_2(T_1(S))$$

So if a cascade of rewrite rules $W_1, \dots, W_n$ can be expressed as a series of FSTs $T_1, \dots, T_n$, there is a single FST T that is a composition $T_1 \circ \dots \circ T_n$ and is equivalent to the cascade $W_1, \dots, W_n$ (Kaplan and Kay, 1994, 364). $G = T_1 \circ \dots \circ T_n$ constitutes a regular grammar. Efficient algorithms are known for compiling a cascade of rewrite rules into an FST (Mohri and Sproat, 1996).

7. AN FST FOR PāṇiniAN SANDHI

The XML formalism for expressing Pāṇinian rules in §3 contains a number of devices; it is not immediately evident how rules employing these devices might be compiled into an FST. A rule compiler, however, is described here that translates Pāṇinian rules expressed in

the XML formalism into rewrite rules that can be automatically compiled into an FST using standard algorithms.

It is useful to begin with an instance of Pāṇinian sandhi derivation of the string *devo 'pi* < *devas api* 'also a god'.

FORM	SŪTRA
`devas api`	
`deva$ api`	8.2.66
`deva#u api`	6.1.113
`dev!(gunate(u)) api`	6.1.87
`devo 'pi`	6.1.109

Sūtra 8.2.66 replaces a *pada*-final *s* with *rU* (here symbolized by `$`). Sūtra 6.1.113 replaces *pada*-final *rU* with *u* preceded by a boundary marker (#) when the next *pada* begins with *a*. Sūtra 6.1.87 has been discussed above. Sūtra 6.1.109 replaces *pada*-initial *a* with *avagraha* (represented by `'` in SLP1) when the previous *pada* ends in the *pratyāhāra eṄ* (i. e. *e* or *o*).

Although the PCREs in the XML format exceed the power of regular relations, and the implementation of mappings and functions is not obvious in a regular grammar, the rule compiler mentioned above is able to produce a cascade of rewrite rules that may be efficiently compiled into an FST. A consequence of the rule compilation strategy is that a single Pāṇinian rule may be represented as several rewrite rules. Where a rule makes use of pattern memory, which can contain n possible values, the rule is expanded into n rewrite rules. Mappings and functions are automatically applied where they occur in ψ (the substituens).

The following cascade represents the four sūtras exemplified above:

s → $ / ____ (␣ | #)
$ → #u / a ____ (␣ |#)a
(a|A)(␣ | #)(a|A) → a
(a|A)(␣ | #)(i|I) → e
(a|A)(␣ | #)(u|U) → o
(a|A)(␣ | #)(f|F) → ar
(a|A)(␣ | #)(x|X) → al
a → ' / (e|o)(␣ | #) ____

Although *avagraha* is not a phoneme, it may be conceived of linguistically as a "trace" left after the deletion of *a*-. For this reason, it is represented in the SLP1 phonological coding.

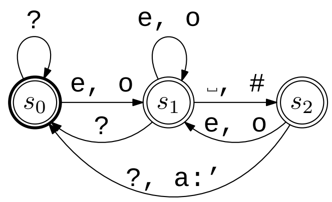


Figure 3: An FST for sūtra 6.1.109

Here the notation $(x|y|z)$ expresses alternation; the rule matches either x, or y, or z. The symbol ␣ represents a space character; the symbol # represents a boundary that has been inserted by a rule.

These rules may be efficiently compiled into FSTs. An FST encoding the final rule is shown in fig. 3. In this transducer, s_0 is the initial state and doubly-circled states are the members of F, the set of final states. The graphical representation of the FST has been simplified, so that symbols that are accepted on transition from s_i to s_j share an arc between s_i and s_j (in this case, the symbols are separated by commas). The notation $x : y$ indicates the symbols on the upper and lower tapes of the transducer, respectively. If the transducer reads input from the upper tape and writes output to the lower tape, $x : y$ indicates that if x is read on the upper tape, y is written on the lower tape. If the symbols on the upper and lower tapes are the same, a shorthand notation is used; thus x is equivalent to $x : x$. The special symbol `?` indicates that *any* symbol is matched on either the upper or lower tape.

Since it is possible to translate each rule in the above cascade into an FST, and FSTs are closed under composition, it is possible to compose a single FST that implements the portion of Pāṇini's sandhi represented by the rules in the cascade above. A compiler developed by the author will be capable of compiling the entirety of Pāṇini's sandhi rules into a single FST. The FST is then converted to Java code using a toolkit written by the author. Compilation of the generated source code yields binary code that may be run portably using any Java Virtual Machine (JVM) (Lindholm and Yellin, 1999). Alternatively, for improved performance, the Java code may be compiled into native machine code using the GNU Compiler for Java (gcj).

8. IMPLICATIONS

While one of the guiding principles of Pāṇini's grammar is conciseness (*lāghava*), a computational implementation poses other demands, such as tractability and efficiency. Pāṇini's rules are formulated in terms of classes based on distinctive features and must be construed with the aid of 'interpretive' (*paribhāṣā*) rules, technical terms (*saṃjñā*), and other theoretical apparatus.

However desirable the mathematical properties of a regular grammar may be, a grammar stated in such terms is at odds with Pāṇini's principles. Rather, it hearkens back to the *vikāra* system employed by earlier linguistic thinkers, in which individual segments are the target of specific rules (Cardona, 1965b, 311).

By way of contrast, Pāṇini states a rule economically in terms of the sound classes enumerated in the *Śivasūtras*. Thus 8.4.41 *ṣṭunā ṣṭuḥ* specifies the retroflexion of an *s* or dental stop either (1) before a *pada*-final -*ṣ* or (2) *pada*-finally before a *ṭU* (i. e. a retroflex stop).

With a little less economy, we can represent Pāṇini's rule by two rules in XML:

```
<rule source="[s@(tu)]"
      target="%(retroflex($1))"
      lcontext="z[@(wb)]"
      ref="A.8.4.41"/>

<rule source="[s@(tu)]"
      target="%(retroflex($1)"
      rcontext="[@(wb)][@(wu)]"
      ref="A.8.4.41"/>
```

The *pratyāhāra tU* stands for the dental stop series {*t*, *th*, *d*, *dh*, *n*}. We apply the retroflex mapping:

```
<mapping name="retroflex">
  <map from="s" to "z"/>
  <map from="@(tu)" to "@(wu)"/>
</mapping>
```

The effect is to change a single phonological feature across an entire class; sounds with a dental place of articulation (*dantya*) are replaced by sounds with a retroflex articulation (*mūrdhanya*). In specifying such a replacement, Pāṇini makes use of the principle of *sāvarṇya* 'homogeneity of sounds'. So the substituend chosen is that closest to the original—with respect to voice (*ghoṣavat / aghoṣa*), aspiration (*mahāprāṇa / alpaprāṇa*), and nasality (*sānunāsika / niranunāsika*). Thus *t* → *ṭ*, *th* → *ṭh*, *d* → *ḍ*, and so on; yet Pāṇini does not need explicitly (and repetitively) to specify the exact segments substituted. In this way, the *Aṣṭādhyāyī* avoids the bias ("segmentalism") that places the linear segment at the center of phonological theory—a bias from which contemporary linguistics is beginning to distance itself (Aronoff, 1992).

The finite state approach discussed in this paper, however, is limited to describing the relations between strings (sequences of segments). As far as the computational model is concerned, individual symbols are atomic, and no class relations between the symbols exist. The goal in this study has been to develop an intermediate representational structure, based on XML, that can faithfully encode some of the linguistically significant aspects of a portion of Pāṇini's grammar (the rules involved in external sandhi) and at the same can be automatically translated into an efficient computational implementation.

9. APPENDIX: CORE RULES FOR EXTERNAL SANDHI

The following is a list of modeled *vidhi* rules (8.3.4: *adhikāra*, 8.4.44: *pratiṣedha*) for external sandhi. No account is taken here of *pragṛhya* rules that specify certain sounds as exempt from sandhi (since these rules refer to morphosyntactic categories). Various optional rules are not listed.

6.1.73	*che ca*
6.1.74	*āṅmāṅośca*
6.1.132	*etattadoḥ sulopo 'koranañsamāse hali*
8.2.66	*sasajuṣo ruḥ*
8.2.68	*ahan*
6.1.113	*ato roraplutādaplute*
6.1.114	*haśi ca*
6.1.101	*akaḥ savarṇe dīrghaḥ*
6.1.88	*vṛddhireci*
6.1.87	*ādguṇaḥ*
6.1.77	*iko yaṇaci*
6.1.109	*eṅaḥ padāntādati*
6.1.78	*eco 'yavāyāvaḥ*
8.2.39	*jhalāṃ jaśo 'nte*
8.3.4	*anunāsikātparo 'nusvāraḥ*
8.3.7	*naśchavyapraśān*
8.3.14	*ro ri*
8.3.17	*bhobhagoaghoapūrvasya yo 'śi*
8.3.15	*kharavasānayorvisarjanīyaḥ*
8.3.19	*lopaḥ śākalyasya*
8.3.20	*oto gārgyasya*
8.3.23	*mo 'nusvāraḥ*
8.3.31	*śi tuk*
8.3.32	*ṅamo hrasvādaci ṅamuṇnityam*
8.3.34	*visarjanīyasya saḥ*
8.3.35	*śarpare visarjanīyaḥ*
8.3.40	*namaspurasorgatyoḥ*
8.4.40	*stoḥ ścunā ścuḥ*
8.4.44	*śāt*
8.4.41	*ṣṭunā ṣṭuḥ*
8.4.45	*yaro 'nunāsike 'nunāsiko vā*
8.4.53	*jhalāṃ jaśjhaśi*
8.4.55	*khari ca*
8.4.60	*torli*
8.4.62	*jhayo ho 'nyatarasyām*
8.4.63	*śaścho 'ṭi*
8.4.65	*jharo jhari savarṇe*

10. REFERENCES

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1988. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, Reading, MA.

Mark Aronoff. 1992. Segmentalism in linguistics: The alphabetic basis of phonological theory. In Pamela Downing, Susan D. Lima, and Michael Noonan, editors, *The Linguistics of Literacy*, volume 21 of *Typological Studies in Language*, pages 71–82. John Benjamins, Amsterdam.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI, Stanford, CA.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1996. *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India, New Delhi.

George Cardona. 1965a. On Pāṇini's morphophonemic principles. *Language*, 41(2):225–237.

George Cardona. 1965b. On translating and formalizing Pāṇinian rules. *Journal of the Oriental Institute, Baroda*, 14:306–314.

George Cardona. 1969. Studies in Indian grammarians: I. The method of description reflected in the Śivasūtras. *Transactions of the American Philosophical Society*, 59(1):3–48.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. MIT Press, Cambridge, MA.

Noam Chomsky. 1956. Three models for the description of language. *IEEE Transactions on Information Theory*, 2(3):113–124.

Ronald M. Kaplan and Matin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):332–378.

Tim Lindholm and Frank Yellin. 1999. *The Java[TM] Virtual Machine Specification*. Addison-Wesley, Reading, MA, 2d edition.

Mehryar Mohri and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 231–238, Santa Cruz, CA. ACL.

Prezemyslaw Prusinkiewicz and Aristid Lindenmayer. 1990. *The Algorithmic Beauty of Plants*. Springer, New York.

Peter M. Scharf and Malcolm D. Hyman. 2007. Linguistic issues in encoding Sanskrit. Unpublished manuscript, Brown University.

Rama Nath Sharma. 1987. *The Aṣṭādhyāyī of Pāṇini, Vol. 1: Introduction to the Aṣṭādhyāyī as a Grammatical Device*. Munshiram Manoharlal, New Delhi.

Rama Nath Sharma. 2003. *The Aṣṭādhyāyī of Pāṇini, Vol. 6: English Translation of Adhyāyas Seven and Eight with Sanskrit Text, Transliteration, Word-Boundary, Anuvṛtti, Vṛtti, Explanatory Notes, Derivational History of Examples, and Indices*. Munshiram Manoharlal, New Delhi.

Henry Smith. 1992. Brevity in Pāṇini. *Journal of Indian Philosophy*, 20:133–147.

Unicode Consortium. 2006. *The Unicode Standard, Version 5.0*. Addison-Wesley, Boston.

Larry Wall, Tom Christiansen, and John Orwant. 2000. *Programming Perl*. O'Reilly, Sebastapol, CA, 3d edition.

Analysis of Sanskrit Text: Parsing and Semantic Relations

Pawan Goyal
Electrical Engineering,
IIT Kanpur,
208016, UP,
India
pawangee@iitk.ac.in

Vipul Arora
Electrical Engineering,
IIT Kanpur,
208016, UP,
India
vipular@iitk.ac.in

Laxmidhar Behera
Electrical Engineering,
IIT Kanpur,
208016, UP,
India
lbehera@iitk.ac.in

ABSTRACT

In this paper, we are presenting our work towards building a dependency parser for Sanskrit language that uses deterministic finite automata(DFA) for morphological analysis and 'utsarga apavaada' approach for relation analysis. A computational grammar based on the framework of Panini is being developed. A linguistic generalization for Verbal and Nominal database has been made and declensions are given the form of DFA. Verbal database for all the class of verbs have been completed for this part. Given a Sanskrit text, the parser identifies the root words and gives the dependency relations based on semantic constraints. The proposed Sanskrit parser is able to create semantic nets for many classes of Sanskrit paragraphs(अनुच्छेद). The parser is taking care of both external and internal sandhi in the Sanskrit words.

1. INTRODUCTION

Parsing is the "de-linearization" of linguistic input; that is, the use of grammatical rules and other knowledge sources to determine the functions of words in the input sentence. Getting an efficient and unambiguous parse of natural languages has been a subject of wide interest in the field of artificial intelligence over past 50 years. Instead of providing substantial amount of information manually, there has been a shift towards using Machine Learning algorithms in every possible NLP task. Among the most important elements in this toolkit are state machines, formal rule systems, logic, as well as probability theory and other machine learning tools. These models, in turn, lend themselves to a small number of algorithms from well-known computational paradigms. Among the most important of these are state space search algorithms, (Bonet, 2001) and dynamic programming algorithms (Ferro, 1998). The need for unambiguous representation has lead to a great effort in stochastic parsing (Ivanov, 2000).

Most of the research work has been done for English sentences but to transmit the ideas with great precision and mathematical rigor, we need a language that incorporates the features of artificial intelligence. Briggs (Briggs,1985) demonstrated in his article the salient features of Sanskrit language that can make it serve as an Artificial language. Although computational processing of Sanskrit language has been reported in the literature (Huet, 2005) with some computational toolkits (Huet, 2002), and there is work going on towards developing mathematical model and dependency grammar of Sanskrit(Huet, 2006), the proposed Sanskrit parser is being developed for using Sanskrit language as Indian networking language (INL). The utility of advanced techniques such as stochastic parsing and machine learning in designing a Sanskrit parser need to be verified.

We have used deterministic finite automata for morphological analysis. We have identified the basic linguistic framework which shall facilitate the effective emergence of Sanskrit as INL. To achieve this goal, a computational grammar has been developed for the processing of Sanskrit language. Sanskrit has a rich system of inflectional endings (vibhakti). The computational grammar described here takes the

concept of vibhakti and karaka relations from Panini framework and uses them to get an efficient parse for Sanskrit Text. The grammar is written in 'utsarga apavaada' approach i.e rules are arranged in several layers each layer forming the exception of previous one. We are working towards encoding Paninian grammar to get a robust analysis of Sanskrit sentence. The paninian framework has been successfully applied to Indian languages for dependency grammars (Sangal, 1993), where constraint based parsing is used and mapping between karaka and vibhakti is via a TAM (tense, aspect, modality) tabel. We have made rules from Panini grammar for the mapping. Also, finite state automata is used for the analysis instead of finite state transducers. The problem is that the Paninian grammar is generative and it is just not straight forward to invert the grammar to get a Sanskrit analyzer, i.e. its difficult to rely just on Panini sutras to build the analyzer. There will be lot of ambiguities (due to options given in Panini sutras, as well as a single word having multiple analysis). We need therefore a hybrid scheme which should take some statistical methods for the analysis of sentence. Probabilistic approach is currently not integrated within the parser since we don't have a Sanskrit corpus to work with, but we hope that in very near future, we will be able to apply the statistical methods.

The paper is arranged as follows. Section 2 explains in a nutshell the computational processing of any Sanskrit corpus. We have codified the Nominal and Verb forms in Sanskrit in a directly computable form by the computer. Our algorithm for processing these texts and preparing Sanskrit lexicon databases are presented in section 3. The complete parser has been described in section 4. We have discussed here how we are going to do morphological analysis and hence relation analysis. Results have been enumerated in section 5. Discussion, conclusions and future work follow in section 6.

2. A STANDARD METHOD FOR ANALYZING SANSKRIT TEXT

The basic framework for analyzing the Sanskrit corpus is discussed in this section. For every word in a given sentence, machine/computer is supposed to identify the word in following structure. $< Word >< Base >< Form >< Relation >$.
The structure contains the root word (<Base>) and its form <attributes of word> and relation with the verb/action or subject of that sentence. This analogy is done so as to completely disambiguate the meaning of word in the context.

2.1. <Word>

Given a sentence, the parser identifies a singular word and processes it using the guidelines laid out in this section. If it is a compound word, then the compound word with सन्धि has to be undone. For example: नदीमागच्छत्=नदीम्+आगच्छत्.

2.2. <Base>

The base is the original, uninflected form of the word. Finite verb forms, other simple words and compound words are each indicated differently. For Simple words: The computer activates the DFA on the ISCII code (ISCII,1999) of the Sanskrit text. For compound words: The computer shows the nesting of internal and external समास using nested parentheses. Undo सन्धि changes between the component words.

2.3. <Form>

The <Form> of a word contains the information regarding declensions for nominals and state for verbs.

- For undeclined words, just write u in this column.
- For nouns, write first.m, f or n to indicate the gender, followed by a number for the case (1 through 7, or 8 for vocative), and s, d or p to indicate singular, dual or plural.
- For adjectives and pronouns, write first a, followed by the indications, as for nouns, of gender (skipping this for pronouns unmarked for gender), case and number.
- For verbs, in one column indicate the class (गण) and voice. Show the class by a number from 1 to 11. Follow this (in the same column) by '1' for parasmaipada, '2' for ätmanepada and '3' for ubhayapada. For finite verb forms, give the root. Then (in the same column) show the tense as given in Table 3. Then show the inflection in the same column, if there is one. For finite forms,

Table 1: Codes for <Form>

pa/	passive
ca/	causative
de/	desiderative
fr/	frequentative

Table 2: Codes for Finite Forms, showing the Person and the Number

1	प्रथम पुरुष
2	मध्यम पुरुष
3	उत्तम पुरुष
s	singular
d	dual
p	plural

Table 3: Codes for Finite verb Forms, showing the Tense

pr	present
if	imperfect
iv	imperative
op	optative
ao	aorist
pe	perfect
fu	future
f2	second future
be	benedictive
co	conditional

show the person and number with the codes given in Table 2. For participles, show the case and number as for nouns.

2.4. <Relation>

The relation between the different words in a sentence is worked out using the information obtained from the analysis done using the guidelines laid out in the previous subsections. First write down a period in this column followed by a number indicating the order of the word in the sentence. The words in each sentence should be numbered sequentially, even when a sentence ends before the end of a text or extends over more than one text. Then, in the same column, indicate the kind of connection the word has to the sentence, using the codes given in table 4.

Then, in the same column, give the number of the other word in the sentence to which this word is connected as modifier or otherwise. The relation set given above is not exhaustive. All the 6 karakas are defined as in relation to the verb.

Table 4: Codes for <Relation>

v	main verb
vs	subordinate verb
s	subject(of the sentence or a subordinate clause)
o	object(of a verb or preposition)
g	destination(gati) of a verb of motion
a	Adjective
n	Noun modifying another in apposition
d	predicate nominative
m	other modifier
p	Preposition
c	Conjunction
u	vocative, with no syntactic connection
q	quoted sentence or phrase
r	definition of a word or phrase(in a commentary)

3. ALGORITHM FOR SANSKRIT RULEBASE

In the section to follow in this paper, we shall explain two of the procedures/algorithms that we have developed for the computational analysis of Sanskrit. Combined with these algorithms, we have arrived at the skeletal base upon which many different modules for Sanskrit linguistic analysis such as: relations, सन्धि, समास can be worked out.

3.1. Sanskrit Rule Database

Every natural language must have a representation, which is directly computable. To achieve this we have encoded the grammatical rules and designed the syntactic structure for both the nominal and verbal words in Sanskrit. Let us illustrate this structure for both the nouns and the verbs with an example each .

Noun:-Any noun has three genders: Masculine,Feminine and Neuter. So also the noun has three numbers: Singular, Dual and Plural. Again there exists eight classification in each number: Nominative, Accusative, Imperative, Dative, Ablative, Genitive, Locative and Vocative. Interestingly these express nearly all the relations between words in a sentence .

In Sanskrit language, every noun is deflected following a general rule based on the ending alphabet such as अकारान्त. For example, राम is in class

अकारान्त which ends with अ(a). Such classifications are given in Table 5. Each of these have different inflections depending upon which gender they correspond to. Thus अकारान्त has different masculine and neuter declensions, आकारान्त has masculine and feminine declensions, इकारान्त has masculine, feminine and neuter declensions. We have then encoded each of the declensions into ISCII code, so that it can be easily computable in the computer using the algorithm that we have developed for the linguistic analysis of any word .

Table 5: attributes of the declension for noun

$Class^*$		$Case^\eta$	$Gender^\varsigma$
अकारान्त(1)	दकारान्त(14)	कर्त्ता(1)	पुल्लिङ(1)
आकारान्त(2)	धकारान्त(15)	कर्म(2)	स्त्रीलिङ(2)
इकारान्त(3)	नकारान्त(16)	करण(3)	नपुंसकलिङ(3)
ईकारान्त(4)	नकारान्त(17^1)	सम्प्रदान(4)	$Number^@$
उकारान्त(5)	पकारान्त(18)	अपादान(5)	एकवचन(1)
ऊकारान्त(6)	भकारान्त(19)	सम्बन्ध(6)	द्विवचन(2)
ऋकारान्त(7)	रकारान्त(20)	अधिकरण(7)	बहुवचन(3)
ऐकारान्त(8)	वकारान्त(21)	सम्बोधन(8)	
ओकारान्त(9)	शकारान्त(22)		
औकारान्त(10)	षकारान्त(23)		
चकारान्त(11)	सकारान्त(24)		
जकारान्त(12)	हकारान्त(25)		
तकारान्त(13)			

Let us illustrate this structure for the noun with an example . For अकारान्त, masculine, nominative, singular declension:

This is encoded in the following syntax: $(163\{1^*, 1^\eta, 1^\varsigma, 1^@\})$.

Where 163 is the ISCII code of the declension (Table 6). The four 1's in the curly brackets represent Class, Case, Gender and Number respectively (Table 5) .

Table 6: Noun example

अ Masculine	Singular(एकवचन)	
	Endings	ISCII Code
Nominative	:	163

Pronouns:-According to Paninian grammar and Kale, (Kale) Sanskrit has 35 pronouns which are: सर्व, विश्व, उभ, उभय, इतर, इतम, अन्य, अन्यतर, इतर, त्वत्, त्व, नेम, यम, यिम, पूर्व, पर, अवर, दक्षिण, उपर, अधर, स्व, अन्तर, त्यद्, एतद्, इतद्, अदस्, एक, द्वि, युष्मद, भवत् and किम् .

We have classified each of these pronouns into 9 classes: Personal, Demonstrative, Relative, Indefinitive, Correlative, Reciprocal and Possessive. Each of these pronouns have different inflectional forms arising from different declensions of the masculine and feminine form. We have codified the pronouns in a form similar to that of nouns .

Adjectives:- Adjectives are dealt in the same manner as nouns. The repetition of the linguistic morphology is avoided .

Verbs:- A Verb in a sentence in Sanskrit expresses an action that is enhanced by a set of auxiliaries"; these auxiliaries being the nominals that have been discussed previously .

The meaning of the verb is said to be both *vyapara* (action, activity, cause), and *phala* (fruit, result, effect). Syntactically, its meaning is invariably linked with the meaning of the verb "to do". In our analysis of Verbs, we have found that they are classified into 11 classes(गण, Table 7). While coding the endings, each class is subdivided according to "इट्" knowledge, सेट्, अनिट् and वेट्; each of which is again sub-classified as into 3 sub-classes as आत्मनेपद, परस्मैपद andउभयपद, which we have denoted as pada. Each verb sub-class again has 10 *lakaaras*, which is used to express the tense of the action. Again, depending upon the form of the sentence, again a division of form as कर्त्तवाच्य, कर्मवाच्य and भाववाच्य has been done. This classification has been referred to as voice. This structure has been explained in Table 7.

.

Let us express the structure via an example for भ्वादिगण, परस्मैपद, Present Tense, First person, Singular. This is encoded in the following syntax: $(219(194\{1^*, 1^\gamma, 2^\eta, 1^\varsigma, 1^\lambda, 1^@, 1^\delta\}))$.
Where 219194 is the ISCII code of the endings (Table 8). The numbers in curly brackets represent class, "it",

Table 7: attributes of the declension for verb

$Class^*$	it^γ	$pada^\eta$	$Tense^\zeta$
भ्वादिगण(1)	सेट्(1)	आत्मनेपद(1)	लट्(1)
अदादिगण(2)	अनिट्(2)	परस्मैपद(2)	लङ(2)
दिवादिगण(3)	वेट्(3)	उभयपद(3)	लृट(3)
स्वादिगण(4)			लोट्(4)
तुदादिगण(5)			विधिलिङ(5)
रुढादिगण(6)			आर्शीलिङ(6)
तनादिगण(7)			लिट्(7)
क्र्यादिगण(8)			लुट्(8)
चुरादिगण(9)			लुङ(9)
जुहोत्यादिगण(10)			लृङ(10)
कण्ड्वादिगण(11)			

$Voice^\lambda$	$Person^@$	$Number^\delta$
कर्त्तवाच्य(1)	प्रथम पुरुष(1)	एकवचन(1)
कर्मवाच्य(2)	मध्यम पुरुष(2)	द्विवचन(2)
भाववाच्य(3)	उत्तम पुरुष(3)	बहुवचन(3)

pada, tense, voice, person and number respectively (Table 7).

Table 8: Verb example

PRESENT	Singular(एकवचन)	
	Endings	ISCII Code
First	ति	219194

Separate database files for nominals and verbs have been maintained, which can be populated as more and more Sanskrit corpsuses are mined for data. The Sanskrit rule base is prepared using the ”Sanskrit Database Maker” developed during this work.

3.2. Deterministic Finite Automata: Sanskrit Rule Base

We have used deterministic finite automata (DFA) (Hopcraft, 2002) to compute the Sanskrit rule base, which we developed as described in section III A. Before we explain the DFA, let us define it.
A deterministic finite automaton consists of:

1. A finite set of states, often denoted Q.
2. A finite set of input symbols, often denoted S.
3. A transition function that takes as arguments a state and input symbol and returns a state, often commonly denoted d.
4. A start state, one of the states in Q, denoted q0.
5. A set of *final* or *accepting states* F. The set F is a subset of Q.

Thus, we can define a DFA in this ”five-tuple” notation: $A = (Q, S, d, q0, F)$. With this short discussion of the DFA, we shall proceed to the DFA structure for our Sanskrit Rule Base. Since we are representing any word by ISCII codes that range from 161 to 234, we have effectively 74 input states. In the notation given below, we are representing the character set by $\{C0, C1, \ldots, C73\}$, where Ci is the character corresponding to the ISCII code $161 + i$. Thus, if we define a DFA = $M(Q, S, d, q0, F)$ for our Sanskrit Rule Database, each of the DFA entities are as follows:

- Q = $\{q_0, q_{C0}, q_{C1}, \ldots, q_{C73}\} \times \{0, 1\}$. 0 represents that the state is not a final state and 1 tells that the state is a final state.
- $\sum = \{C0, C1, \ldots, C73\}$
- $\delta((q_x, a), Y) = \delta(q_Y, a)$or $\delta(q_Y, b)$ a,b $\epsilon\{0, 1\}$
- $q_0 =< q_0, 0 >$
- $F \subset \{q_{C0}, q_{C1}, \ldots, q_{C73}\} \times \{1\}$

In this work, we have made our DFA in a matrix form with each row representing the behavior of a particular state. In a given row, there are 74 columns and entries in a particular column of the corresponding row store the state we will finally move to on receiving the particular input corresponding to the column. In addition, each row carries the information whether or not it is a final state.
For example: $D[32][5] = 36$ conveys that in the DFA matrix D[i][j], in 32nd state, if input is $C5$, we will move to state no. 36 .(To be noted: $C5$ is the character corresponding to the ISCII code 166.).

In the graph below, we are giving an example how the DFA will look as a tree structure. The particular graph is constructed for the verb declensions for the class भ्वादिगण. The pada is परस्मैपद and the tense is present tense. The search in this DFA will be as follows:- If the first ending of the input corresponds to one of the state 163, 195 or 219, we will move ahead

in the DFA otherwise the input is not found in this tree. On getting a match, the search will continue in the matched branch. .

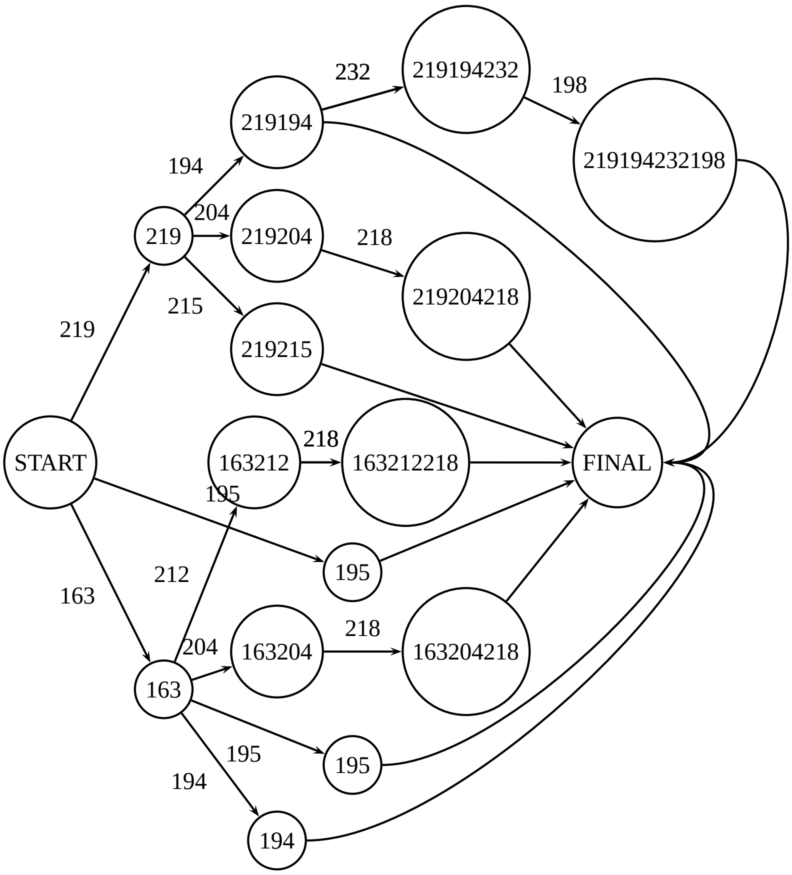


Figure 1: DFA tree obtained for भ्वादिगण परस्मैपद present tense.

In general, the search in the DFA is done as follows (We take the example of searching for भवथः in the DFA tree constructed above:-

- Firstly, an input word is given as the input to the user interface in Devanagari format .
- The word is changed to its equivalent ISCII code (203212195163 in this case).
- The automaton reads the forms in the reverse order to lemmatize them. In our DFA, we give one by one the last three digits of the ISCII code till the matching is there.
  - Start state: 000
  - input to DFA: 163, i.e character $C2$.
  - In the DFA matrix we will check the entry D[0][2]. If it is zero, no match is there for this entry and hence no match either for the word. Else we will move to the state specified by the entry.
  - In this case, we get the entry corresponding to state "163". That means it is either an intermediate state or a final state. From the graph, it is visible that the tree accepts 163 just after the start state. Also, it is not a final state. Now we will have 195 (i.e. $C34$) as next input and 34^{th} column of the row corresponding to state 232 will be checked and the search continues till no match.
  - Final match will be 163195.
- The final match will be checked for being eligible for a final state which is true in this case. We can verify it from the graph given.
- Remaining part of the word is sent to database engine of program to verify and to get attributes. The word corresponding to the stem, Devanagari equivalent of 203212, that is भव) will be sent to database.
- If both criteria are fulfilled (final state match and stem match through database), we will get the root word and its category (verb in this case). The attributes such as tense, form, voice, class, pada, person, number are coded in the final state itself according to the notations given in table 5 and 7. All the possible attributes are stored and it is left up to the final algorithm to come up with the most appropriate solution.

Let me just explain how we have obtained the deterministic finite automata. Clearly, the states are obtained via input symbols. Ambiguity remains in $\{0, 1\}$. If the state is not a final state at all, it is declared as intermediate state without any ambiguity to be considered for non-deterministic. When the state is a final state, for example consider गच्छति and गमिष्यति. When we encounter ति in गच्छति, we get the root as गच्छ्. (Of course, we have to add the हलन्त and go through धात्वादेश getting गम् as root verb.) But in गमिष्यति, ति is not a final state. It seems at this point that we could have obtained a non-deterministic finite

automaton. We have resolved the problem by accepting the following facts:

1. Final state can be intermediate state too but not the other way round.

2. Our algorithm doesn't stop just as it gets to a final state, it goes to the highest possible match, checks it for being final state and, in case it isn't, it backtracks and stops at the optimal match which satisfies the two criteria as told in the algorithm (final state match and stem match through database).

There might be another ambiguity too, for example, in रामाभ्याम्, आभ्याम् is a final state but it refers to करण, सम्प्रदान, and अपादान karaka. This seems to be non-deterministic. We have avoided this problem by suitably defining the states. Final state represents all possibilities merged in a single state. It is up to the algorithm to come up with the unique solution. There could be situation where longest match is not the right assignment. To deal with this, all other possible solutions are also stacked and are substituted (if needed) when we go for relation analysis. For example, let us take the word सनाभ्याम्. We assume that सना, सनाभि and सनाभ्या are valid root words. Our algorithm will choose सना as root word along with the attributes (3 possibilities here). But the other solutions are also stacked in decreasing order of the match found. It is discussed in the relation analysis, how we deal with this situation.

4. ALGORITHM FOR SANSKRIT PARSER

The parser takes as input a Sanskrit sentence and using the Sanskrit Rule base from the DFA Analyzer, analyzes each word of the sentence and returns the base form of each word along with their attributes. This information is analyzed to get relations among the words in the sentence using If-Then rules and then output a complete dependency parse. The parser incorporates Panini framework of dependency structure. Due to rich case endings of Sanskrit words, we are using morphological analyzer. To demonstrate the Morphological Analyzer that we have designed for subsequent Sanskrit sentence parsing, the following resources are built:

- Nominals rule database (contains entries for nouns and pronouns declensions)
- Verb rule database (contains entries for 10 classes of verbs)
- Particle database (contains word entries)

Now using these resources, the morphological analyzer, which parses the complete sentences of the text is designed.

4.1. Morphological Analysis

In this step, the Sanskrit sentence is taken as input in Devanagari format and converted into ISCII format. Each word is then analyzed using the DFA Tree that is returned by the above block. Following along any path from start to final of this DFA tree returns us the root word of the word that we wish to analyze, along with its attributes. While evaluating the Sanskrit words in the sentence, we have followed these steps for computation:

1. First, a left-right parsing to separate out the words in the sentence is done.

2. Second, each word is checked against the Sanskrit rules base represented by the DFA trees in the following precedence order: Each word is checked first against the avavya database, next in pronoun, then verb and lastly in the noun tree.

The reason for such a precedence ordering is primarily due to the fact that *avavya* and *pronouns* are limited in number compared to the verbs, and verbs are in-turn limited compared to the infinite number of nouns that exist in Sanskrit.

4.1.1. Sandhi Module

In the analysis, we have done, the main problem was with words having external sandhi. Unless we are able to decompose the word into its constituents, we are unable to get the morph of the word. So, a rulebase sandhi analyzer is developed which works on the following principles.

- Given an input word, it checks at each junction for the possibility of sandhi.
- If it finds the junction, it breaks the word into possible parts and sends the first part in the DFA.
  - If it finds a match, it sends the second part in DFA.

    * If no match, it recursively calls the sandhi module (For the possibility of multiple sandhi in a single word).
    * If match is found, terminates and returns the words.
  - If no match, it goes to the next junction.

The rules for decomposing the words are taken from Panini grammar. The search proceeds entirely backwards on the syllabic string. Emphasis is given on minimum possible breaks of the string, avoiding overgeneration.
Panini grammar has separate sections for vowel sandhi as well as consonant sandhi. Also, there is specification of visarga sandhi. Below, we are describing the simplified rules for undoing sandhi.

Vowel Sandhi:- We have considered दीर्घ संधि, वृद्धि संधि, गुण संधि, यण संधि and अयादि संधि in vowels. (पररूप, पूर्वरूप and प्रकृतिभाव are not taken into account yet.)

1. दीर्घ संधि:- If the junction is the मात्रा corresponding to आ, ई, ऊ or ॠ, it is a candidate for दीर्घ संधि. The algorithm for an example word भानूदय is explained.

   - We assume that we don't get any match at the junction आ after भ.
   - The junction ऊ is a candidate for दीर्घ संधि. So the following breaks are made:
     1. भानु+ उदय, 2. भानु+ ऊदय, 3. भानू+उदय, 4. भानू+ ऊदय. For each break, the left hand word is first sent to DFA and only if it is a valid word, right word will be sent. In this case, first solution comes to be the correct one.

2. वृद्धि संधि:- In this case, the junction is ऐ, औ. The corresponding break-ups are:

   - ऐ:- (अ or आ) + (ए or ऐ).
   - औ:- (अ or आ) + (ओ or औ).

   The algorithm remains the same as told in previous case.

3. गुण संधि:- In this case, the junction is ए, ओ, अर्, अल्. The corresponding break-ups are:

   - ए:- (अ or आ) + (इ or ई).
   - ओ:- (अ or आ) + (उ or ऊ).
   - अर्:- (अ or आ) + (ऋ or ॠ).
   - अल्:- (अ or आ) + (ऌ or ॡ).

   The algorithm follows the same guidelines.

4. यण संधि:- In this case, the junction is a halanta followed by य, व, र, ल. The corresponding break-ups are:

   - halanta + य:- (इ or ई) + अ.
   - halanta + व:- (उ or ऊ) + अ.
   - halanta + र:- (ऌ or ॡ) + अ.
   - halanta + ल:- (ऋ or ॠ) + अ.

   The algorithm follows the same guidelines.

5. अयादि संधि:- In this case, the junction is अय्, आय्, अव्, आव् followed by any vowel. The corresponding break-ups are:

   - अय् + vowel:- ए + vowel. (same vowel is retained.)
   - आय् + vowel:- ऐ + vowel.
   - अव् + vowel:- ओ + vowel.
   - आव् + vowel:- औ + vowel.

   The algorithm follows the same guidelines.

Consonant Sandhi:- For dealing with consonant sandhi, we have defined some groups taking clue from panini grammar such as कु, चु, टु, तु, पु each of which have 5 consonants which are similar in the sense of place of pronunciation. Also, there is a specific significance of first, second, third etc. letter of a specific string. The following ruleset is made:

- Define string s1, with first five entries of टु and 6th entry as ष. Also, define s2, with first five entries of तु and 6th entry as स. The rule says,
  The junction is $a + halanta + c$, and the breakup will be $b + halanta$ and c, where $a, c \epsilon$ s1, $b \epsilon$ s2 and the position of a and b are same in the respective strings.
  For example, in the word रामष्षष्ठः, the junction is ष + halanta + ष. The break-up will be, स +halanta and ष. Hence we get रामस् + षष्ठः.

- Define string s1, with first five entries of चु and 6th entry as श. Also, define s2, with first five entries of तु and 6th entry as स. The rule says,
  The junction is $a + halanta + c$, and the breakup will be $b + halanta$ and c, where $a, c\ \epsilon$ s1, $b\ \epsilon$ s2 and the position of a and b are same in the respective strings.
  For example, in the word सज्जन, the junction is ज + halanta + ज. ज is the third character of string $s1$. The break-up will be, द +halanta and ज. Hence we get सद् + जन.

- We have defined strings as घोष and अघोष with अघोष containing first two characters of all the five strings कु, चु, टु, तु, पु as well as श, ष, स. घोष contains all other consonants and all the vowels. The rule says, if we get a junction with $a + halanta + c$, where a,c ϵ घोष, a will be changed to corresponding अघोष while undoing the sandhi. Similarly, other rules are made.

- The vowels are categorized into ह्रस्व and दीर्घ categories. ह्रस्व contains अ, इ, उ, ऋ and दीर्घ contains आ, ई, ऊ, ॠ. If the junction is a + न + halanta + न, where $a\ \epsilon$ ह्रस्व, the break-up will be: a + न + halanta and ϕ, where ϕ denotes null, i.e. other न is removed. For example, तस्मिन्नरण्ये breaks up into तस्मिन् and अरण्ये.

Visarga Sandhi:- We have looked at visarga sandhi in a single word. The rules made are as follows:

- The junction is श + halanta + a ϵ चु. The break-up will be ः and a.

- The junction is ष + halanta + a ϵ टु. The break-up will be ः and a.

- The junction is स + halanta + a ϵ तु. The break-up will be ः and a.

- The junction is र + halanta + a ϵ consonant. The break-up will be ः and a.

- The junction is र + halanta + a ϵ vowel. The break-up will be ः and a.

4.2. Relation Analysis

With the root words and the attributes for each word in hand for the previous step, we shall now endeavor to compute the relations among the words in the sentence. Using these relation values we can determine the structure of each of the sentences and thus derive the semantic net, which is the ultimate representation of the meaning of the sentence.
For computing the relations, we have employed a case-based approach i.e., nominals were classified as subject, object, instrument, recipient (beneficiary), point of separation (apaadaana) and location, to the verb based on the value of the case attribute of the word, as explained under noun example in Section 3.1.

The Sanskrit language has a dependency grammar. Hence the karaka based approach is used to obtain a dependency parse tree. There are reasons for going for dependency parse:

1. Sanskrit is free phrase order language. Hence, we need the same parse for a sentence irrespective of phrase order.

2. Once the karaka relations are obtained, it is very easy to get the actual thematic roles of the words in the sentence.

The problem comes when we have many possible karakas for a given word. We need to disambiguate between them. We have developed some If-Then rules for classifying the nouns, pronouns, verb, sub-verbs and adjectives in the sentence. The rules are as follows: First we are looking at the sentences having at least one main verb. Nominal sentences are to be dealt in the similar manner but the description will be given later.

1. If there is a single verb in the sentence, declare it as the main verb.

2. If there are more than one verb,
   (a) The verbs having suffix क्त्वा, ल्यप्, तुमुन् are declared subverbs of the nearest verb in the sentence having no such affix.
   (b) All other verbs are main verbs of the sentence and relations for all other words are given in regard to the first main verb.

3. For the nouns and pronouns, one state may have many possibilities of the cases. These ambiguities are to be resolved. The hand written rules for determining these ambiguities are as follows

(Rules are written for nouns. Adjective precede nouns (May not precede too due to free word order nature.) and hence get the same case as nouns. For pronouns, rules are same as that for nouns.):

(a) Nominative case: The assumption is that there is only one main subject in an active voice sentence. We proceed as follows:
- All the nouns having nominal case as one of the attributes are listed. (For example, फलम् has both possibilities of being nominative or accusative case.)
- All those connected by च are grouped together and others are kept separate. We now match each group along the following lines:
  - The number matches with that of the verb(Singular/dual/plural).
  - The root word matches with the person of the verb(i.e root word ”अस्मद्” for 3rd person, ”युष्मद्” for 2nd person).

  If ambiguity still remains, the one having masculine/feminine as gender is preferred for being in कर्ता karaka and declared as subject of the main verb.

In passive voice,
- Nominative case is related to main verb as an object. After grouping and going through the match, the noun is declared as object of main verb.

(b) Accusative case: Assuming that the disambiguation for nominative case works well, there is no disambiguation left for this case. All those left with accusative case indeed belong to that. The noun is declared as object to nearest sub-verb or main verb.

(c) Instrumental case: If the sentence is in passive voice, the noun is declared as subject of the main verb.

For active voice, ambiguity remains if the number is dual. The folowing rules are used:
- We seek if the indeclinable such as सह, साकम्, सार्धम्, समम् follow the noun. In that case, noun is declared as instrument.
- If the noun is preceded by time or distance measure or is itself one of these, it is declared as instrument. For example द्वाभ्याम् दिनाभ्याम् नीरोगः जातः, here द्वाभ्याम् is the disambiguating feature.
- If बधिरः, काणः are following noun, the noun is declared as instrumental.

(d) Dative case: For dative case, disambiguity is with respect to ablative case in terms of dual and plural nnumbers The disambiguating feature used here is main verb. That is, there are certain verbs which prefer dative case and certain verbs prefer ablative. For example:
- The verbs preferring dative case are क्रुध्, द्रुह्, ईर्ष्य्, असूय्, स्पृह् etc.
- The verbs preferring ablative case are जुगुप्सा, विराम, प्रमाद, वार etc.

Initially, we have populated the list using अष्टाध्यायी knowledge as well as some grammar books but this has to be done statistically using corpus analysis.

(e) Ablative case: The ambiguity here is for certain nouns with the genitive case in singular person. The ambiguity resolution proceeds along the following lines:
- If the noun having ambiguity has a verb next to it, it will be taken as ablative (Noun with genitive case marker is not followed by a verb.)
- If suffixes तरप्, तमप् are used in the sentence, the noun is declared as ablative.
- If तुल्य, सदृश are following the noun, it is declared as genitive.
- Finally, we look for the disambiguating verbs as done in previous case.

(f) Genitive case: The ambiguity is there in dual with respec to locative case. We have used that by default, it will be genitive since we have not encountered any noun with locative case and dual in number.

(g) Locative case: The ambiguities are already resolved.

Only problematic case will be the situation discussed in section 3.2 with the example of सनाभ्याम्. If the algorithm is able to generate a parse taking the longest possible match, we will not go into stacked possibili-

ties, but if the subject disagrres with the verb (blocking), or some other mismatch is found, we will have to go for stacked possibilities.
Thus, we have got the case markings. Relation for nominative and accusative case markings have already been defined. For other case markings,

- Instrumental: related as an instrument to main verb in certain cases (taken from अष्टाध्यायी).
- Dative: related as recipient to main verb in certain cases, but also denotes the purpose.
- Ablative: related as separation point.
- Genitive: this is not considered as karaka since karaka has been defined as one which takes role in getting the action done. Hence it is related to the word following it.
- Locative: related as location to the main verb.

Still, we have not given any relation to adjectives and adverbs. For each adjective, we track the noun it belongs to and give it the same attributes. It is defined as adjective to the noun. The adverbs are related to the verb it belongs as adverb.

Based on these relations, we can obtain a semantic net for the sentence with verb as the root node and the links between all the nodes are made corresponding to relations with the verb and interrelations obtained.
Sanskrit has a large number of sentences which are said to be nominal sentences, i.e. they don't take a verb. In Sanskrit, every simple sentence has a subject and a predicate. If the predicate is not a finite verb form, but a substantive agreeing with the subject, the sentence is a nominal sentence. In that case, the analysis that we have done above seems not to be used as it is. But in Sanskrit, there is a notion called आक्षेप, that is, if one of the verb or subject is present, other is obtained to a certain degree of definiteness. Take for example, the sentence अहम् कलमेन लिखामि।. If instead of saying the full sentence, I say अहम् कलमेन, लिखामि is determined as verb. Similarly, if I say कलमेन लिखामि, the subject अहम् is determined. अहम् is a kind of appositive expression to the inflectional ending of the verb लिखामि. We have used this concept for analyzing the nominal sentences. That is, verb is determined from the subject. Mostly, the forms of अस् only are used and relations are defined with respect to that. Although, the analysis done is not exhaustive, some ruleset is built to deal with them. Most of the times, relations in a nominal sentence are indicated by pronouns, adjectives, genitive. For example, in the sentence सः सुन्दरः बालकः, there is आक्षेप of the verb अस्ति in the sentence by the subject बालकः. Hence बालकः is related to the verb as subject. सः is a pronoun referring to बालकः and सुन्दरः is an adjective referring to बालकः. Similarly, इदम् तस्य गृहम्।. In this sentence, इदम् is a pronoun referring to गृहम् and तस्य is a genitive to गृहम्. Here again, there will be आक्षेप of the verb अस्ति and गृहम् will be related to the verb as subject.

5. RESULTS

5.1. Databases Developed

The following Sanskrit Rule Databases have been developed during the project:-

- Nominals (शब्दरूप) rule database contains entries for nouns and pronouns declensions along with their attributes.
- Verb (धातुरूप) rule database contains entries for 10 classes of verb along with their tenses.
- Particle (अव्यय) database.

Along with these databases, we have developed some user interfaces (GUI) to extract information from them. For example, if we want to get the forms of a particular verb in a particular tense, we can just open this GUI and give also obtained. the root word and tense information.

5.2. Parser Outputs

Currently, our parser is giving an efficient and accurate parse of Sanskrit text. Samples of four of the paragraphs which have correctly been parsed are given below along with snapshot of one sentences per paragraph.
अनुच्छेद 1:- पूज्याः गुरुचरणाः रमणीये गुरुकुले प्रातः नियमेन सन्ध्याम् उपास्य मेधाविनः शिष्यान् सम्यक् अध्यापयन्ति। गुरुकुले अनेकेछ्हराः सन्ति। केचन बालाः। केचन प्रौढाः। गुरुचरणाः प्रौढान् छात्रान्

पाठयन्ति। प्रौढाः बालान् पाठयन्ति। सर्वे वेदमन्त्रान् विशुद्धरूपेण उच्चारयन्ति। गुरुचरणाः शिष्यान् वेदस्य अर्थम् अपि बोधयन्ति। प्रतिदिनम् प्रातः सायम् सर्वे शिष्याः यज्ञम् अपि कुर्वन्ति। ते वनम् गत्वा समिधः आनयन्ति। ते नियमेन व्यायामम् कुर्वन्ति। एते शिष्याः एव धर्मस्य रक्षणम् करिष्यन्ति।.

S.no.	Word	Root	Group	Attribute	Relation
1	पूज्याः	पूज्य	Adjective	m8p m1p	1 adjective 2
2	गुरुचरणाः	गुरुचरण	Noun	m8p m1p	2 subject 12
3	रमणीये	रमणीय	Adjective	n8d n7s n2d n	3 adjective 4
4	गुरुकुले	गुरुकुल	Noun	n8d n7s n2d n	4 location 12
5	प्रातः	प्रातः	Avyaya		5 adverb 8
6	नियमेन	नियमेन	Avyaya		6 adverb 8
7	सन्ध्याम्	सन्ध्या	Noun	f2s	7 object 8
8	उपास्य	आस्	sub-verb	उप,क्त्वा	8 sub-verb 12
9	मेधाविनः	मेधाविन्	Adjective	m8p m6s m5s	9 adjective 10
10	शिष्यान्	शिष्य	Noun	m2p	10 object 12
11	सम्यक्	सम्यक्	Avyaya		11 adverb 12
12	अध्यापयन्ति	इंड	Verb	pr1p pr1p pr1p	12 verb 12

Figure 2: Parser output for पूज्याः गुरुचरणाः रमणीये गुरुकुले प्रातः नियमेन सन्ध्याम् उपास्य मेधाविनः शिष्यान् सम्यक् अध्यापयन्ति।

अनुच्छेद 2:- मम गृहे एकम् उपवनम् वर्तते। उपवनस्य समीपे एका गोशाला वर्तते। अहम् प्रतिदिनम् प्रातः सूर्योदयात् पूर्वम् शय्याम् त्यक्त्वा शौचादिकम् विधाय गोशालाम् गच्छामि। तत्र मम गौः माम् प्रतीक्षते। मम गोः नाम धवला अस्ति। धवलायाः वत्सायाः नाम गौरी अस्ति। अहम् धवलाम् प्रेम्णा स्पृशामि। तस्याः सास्नायाम् कण्डूये। सा प्रसन्ना भवति। गौरीम् अपि कराभ्याम् संवाहयामि। तत्पश्चात् गोशालाम् मार्जयामि। गोमयम्, भुक्तावशिष्टानि तृणानि च निरस्यामि। ततः मम धवलायै सुस्वादूनि कोमलानि तृणानि ददे। अनन्तरम् गौरीम् मुञ्चामि। सा झटिति स्वमातरम् धवलाम् धयते। धवलायाः ऊधसि क्षीरम् अवतरति। तत्पश्चात् अहम् धवलाम् दुग्धम् दुहे। एतत् मम नित्यकर्मः। गोसेवाकाले अहम् गोमहिमप्रतिपादकान् वेदमन्त्रान् मन्दम् मन्दम् मधुरस्वरेण उच्चारयामि।.

अनुच्छेद 3:- मम गृहे एकम् उपवनम् वर्तते। उपवनम् विशालम् अस्ति। अस्मिन् उपवने नाना फलवृक्षाः सन्ति। बहवः पुष्पतरवः अपि वर्तन्ते। अनेकाः लताः अपि विद्यन्ते। आयुर्वैदिक वनस्पतयः अपि सन्ति। अहम् प्रतिदिनम् सायंकाले उद्यानकर्म कुर्वे। अहम्

S.no.	Word	Root	Group	Attribute	Relation
1	अहम्	अस्मद्	Pronoun	g1s	1 subject 11
2	प्रतिदिनम्	प्रतिदिनम्	Avyaya		2 adverb 7
3	प्रातः	प्रातः	Avyaya		3 adverb 7
4	सूर्योदयात्	सूर्योदय	Noun	m5s	4 separation 5
5	पूर्वम्	पूर्व	Pronoun	n2s m2s	5 object 7
6	शय्याम्	शय्या	Noun	f2s	6 object 7
7	त्यक्त्वा	त्यज्	sub-verb	N,क्त्वा	7 sub-verb 11
8	शौचादिकम्	शौचादिक	Noun	n2s n1s	8 object 9
9	विधाय	धा	sub-verb	वि,क्त्वा	9 sub-verb 11
10	गोशालाम्	गोशाला	Noun	f2s	10 object 11
11	गच्छामि	गम्	Verb	pr3s pr3s pr3s	11 verb 11

Figure 3: Parser output for अहम् प्रतिदिनम् प्रातः सूर्योदयात् पूर्वम् शय्याम् त्यक्त्वा शौचादिकम् विधाय गोशालाम् गच्छामि।

हानिकराणि तृणानि निराकरोमि। अहम् वारिणा सर्वान् वनस्पतीन् सिञ्चामि। सायंकाले उपवने मम पितृचरणाः भ्रमन्ति। तैः साकम् मम भ्रातृजाः अपि भ्रमन्ति। कदाचित् गौरी अपि तत्र आयाति। सः इतस्ततः वेगेन धावति। शाद्वले हरिततृणानि चरति। मयूराः अपि तत्र आगत्य नृत्यन्ति। ते मधुरम् ध्वनिम् कुर्वते। वृक्षेषु वानराः क्रीडन्ति। विविधाः पक्षिणः वृक्षेषु कूजन्ति। मम मातृचरणाः सर्वेभ्यः रोटिकाखण्डानि धान्यकणान् च वितरन्ति। इदम् दृश्यम् अति आनन्दप्रदम् भवति।

S.no.	Word	Root	Group	Attribute	Relation
1	मम	अस्मद्	Pronoun	g6s	1 relation 2
2	मातृचरणाः	मातृचरणा	Noun	f8p f2p f1p	2 subject 7
3	सर्वेभ्यः	सर्व	Pronoun	n5p n4p m5p r	3 recipent 7
4	रोटिकाखण्डानि	रोटिकाखण्ड	Noun	n8p n2p n1p	4 object 7
5	धान्यकणान्	धान्यकण	Noun	m2p	5 object 7
6	च	च	Avyaya		6 adverb 7
7	वितरन्ति	तृ	Verb	pr1p pr1p pr1p	7 verb 7

Figure 4: Parser output for मम मातृचरणाः सर्वेभ्यः रोटिकाखन्डानि धान्यकणान् च वितरन्ति

अनुच्छेद 4:- मम देशस्य नाम भारतवर्षम्। मम देशे वेदानाम् अध्ययनम् भवति। वेदेषु प्राणिनाम् मनुष्याणाञ्च जीवनाय आवश्यकाः नियमाः वर्णिताः सन्ति। वेदेषु अन्यतमः ऋग्वेदः। ऋग्वेदे एकम् वाक्यम् वर्तते,"हे मानव! त्वम् कृषिम् अवश्यम् कुरु।" सम्पूर्णे जगति जन्तूनाम् जीवनाधारः अन्नमेव।अन्नञ्च कृषिम् विना नैव सम्भवति। अतः भारतीयाः आचार्याः सर्वेषाम् जीवानाम् कल्याणाय कृषिकर्मशिक्षणाय बहुविधान् उपायान् अशिक्षयन्त। तेषु प्रधानः उपायः

गोवंशसेवा। गोः वंशे धेनवः, बलीवर्दाः वृषभाः, वत्साः, वृद्धाः धेनवः सर्वे गोवंशजाः अन्तर्भवन्ति। भूमेः बलवर्धनाय गोमूत्रम् गोमयम् च अत्यन्तम् उपकुरूतः। बलदाः भूमिम् कर्षितुम् उपयुज्यन्ते। शस्यानाम् नयनानयनाभ्याम् गोमयजन्यउर्वरकाणाम् इतरपदार्थानाम् इतस्ततः प्रापणाय शकटानाम् प्रयोगः आवश्यकः। शकटाश्च वृषभैरेव उह्यन्ते। अतः गोवंशस्य कृषेश्च अविच्छेद्य कश्चन अपूर्वः सम्बन्धः वर्तते। अतएव गोः उपमा दातुम् न शक्यते। भूमेः एकम् नाम गौः इत्यपि विद्यते। संस्कृते गौः इति पदेन स्त्री गौ पुङ्गवश्च उभावपि गृह्यते। कृषिवर्धनाय शस्यनाशकानाम् कीटानाम् नाशार्थम् केचन घातकद्रव्यानाम् प्रयोगम् कुर्वते। सामवेदस्य प्रथमे आर्चिके कृषिविषये गम्भीरा चर्चा प्राप्यते। अस्माकम् गुरवः महर्षयः सम्पूर्ण जगतः कल्याणाय कृषिविद्याम् सम्यक् प्रचारयन्। कृषिकर्मणे जनान् प्रेरितवन्तः। वेदेषु भूमिः माता इति कथ्यते। यथा माता अस्मान् पालयति तथा पृथ्वी सर्वान् जीवान् अन्नेन फलैः नानाविधपदार्थैः पालयति। यन्त्रैः, तैलेन्धनेन, रसायनद्रव्यैः कीटनाशकैः च या कृषि क्रियते तेन सर्वेषाम् जीवानाम् नाश एव भवति। अनेन कर्मणा सर्वेषाम् प्राणिनाम् मनुष्याणाञ्च महती हानि जायते।.

S.no.	Word	Root	Group	Attribute	Relation
1	वेदेषु	वेद	Noun	m7p	1 location 9
2	प्राणिनाम्	प्राणिन्	Noun	m6p	2 relation 5
3	मनुष्याणाम्	मनुष्य	Noun	m6p	3 relation 5
4	च	च	Avyaya		4 adverb 9
5	जीवनाय	जीवन	Noun	m4s	5 recipient 7
6	आवश्यकाः	आवश्यक	Adjective	m8p m1p	6 adjective 7
7	नियमाः	नियम	Noun	m8p m1p	7 subject 9
8	वर्णिताः	वर्णित	Adjective	m8p m1p	8 adjective 7
9	सन्ति	अस्	Verb	pr1p pr1p pr1p	9 verb 9

Figure 5: Parser output for वेदेषु प्राणिनाम् मनुष्याणाञ्च जीवनाय आवश्यकाः नियमाः वर्णिताः सन्ति।

The parse results pave the way for representing the sentence in the form of a Semantic Net. We here give the semantic net for the parse output given in Figure 2.The Semantic Net is shown in Figure 6.

6. CONCLUSIONS AND FUTURE WORK

Our parser has three parts. First part takes care of the morphology. For each word in the input sentence, a dictionary or a lexicon is to be looked up, and

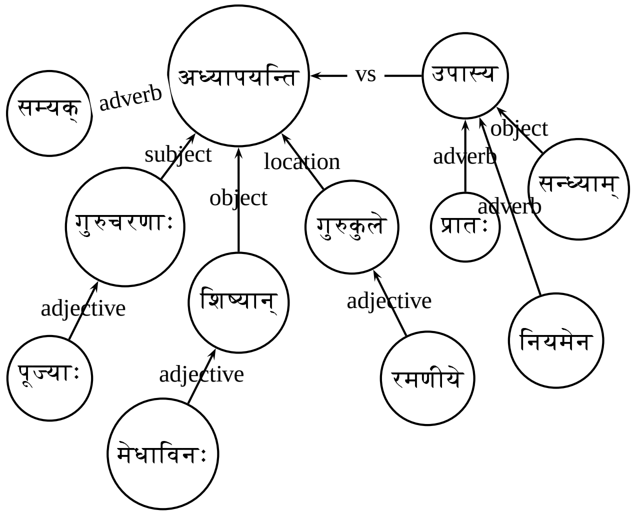


Figure 6: Semantic net representation of the sentence [पूज्याः गुरुचरणाः रमणीये गुरुकुले प्रातः नियमेन सन्ध्याम् उपास्य मेधाविनः शिष्यान् सम्यक् अध्यापयन्ति।]

associated grammatical inforation is retrieved. One of the criterion to judge a morphological analyzer is its speed. We have made a linguistic generalization and declensions are given the form of DFA, thereby increasing the speed of parser. Second part of the parser deals with making ”Local Word Groups”. As noted by Patanjali, any practical and comprehensive grammar should be written in ’utsarga apavaada’ approach. In this approach rules are arranged in several layers each forming an exception of the previous layer. We have used the ’utsarga apavaada’ approach such that conflicts are potentially taken care of by declaring exceptions. Finally, words are grouped together yielding a complete parse. The significant aspect of our approach is that we do not try to get the full semantics immediately, rather it is extracted in stages depending on when it is most appropriate to do so. The results we have got are quite encouraging and we hope to analyze any Sanskrit text unambiguously.

To this end, we have successfully demonstrated the parsing of a Sanskrit Corpus employing techniques designed and developed in section 2 and 3. Our analysis of the Sanskrit sentences in the form of morphological analysis and relation analysis is based on sentences as shown in the four paragraphs in previous section. The algorithm for analyzing compound words is tested separately. Hence future

works in this direction include parsing of compound sentences and incorporating Stochastic parsing. We need to take into account the नामधातु as well. We are trying to come up with a good enough lexicon so that we can work in the direction of समास विच्छेद in Sanskrit sentences. Also, we are working on giving all the rules of Panini the shape of multiple layers. In fact, many of the rules are unimplementable because they deal with intentions, desires etc. For that, we need to build an ontology schema. The Sandhi analysis is not complete and some exceptional rules are not coded. Also, not all the derivational morphology is taken care of. We have left out many प्रत्यय. Reason behind not incorporating the प्रत्यय was that it is difficult to come up with a general DFA tree for any of the प्रत्यय because of the wide number of rules applicable. For that, we need to encode the Panini grammar first.

Acknowledgment

We humbly acknowledge our gratitude to revered Aacharya Sanskritananda Hari, founder and director of Kaushalya pitham Gurukulam, Vadodara for educating us in all aspects of Sanskrit language.

7. REFERENCES

Blai Bonet and Hctor Geffner 2001. *Planning as heuristic search*. Artificial Intelligence 129.

Ferro, M.V., Souto, D.C., Pardo, M.A.A.. 1998. *Dynamic programming as frame for efficient parsing*. Computer science, 1998.

Ivanov, Y.A., Bobick, A.F. 2000. *Recognition of visual activities and interactions by stochastic parsing*. Volume 22, Issue 8, Aug. 2000 Page(s):852 - 872. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Briggs, Rick. 1985. *Knowledge Representation in Sanskrit and artificial Intelligence*, pp 33-39. The AI Magazine.

G. Huet. 2002. *The Zen Computational Linguistics Toolkit*. ESSLLI 2002 Lectures, Trento, Italy.

G. Huet. 2005. *A Functional Toolkit for Morphological and Phonological Processing, Application to a Sanskrit Tagger*. Journal of Functional Programming 15 (4) pp. 573–614.

G. Huet. 2006. *Shallow syntax analysis in Sanskrit guided by semantic nets constraints*. International Workshop on Research Issues in Digital Libraries, Kolkata. Proceedings to appear as Springer-Verlag LNCS, 2007.

Bureau of Indian Standards. 1999. *ISCII: Indian Script Code for Information Interchange*. ISCII-91.

Akshar Bharati and Rajeev Sangal. 1993. *Parsing Free Word Order Languages in the Paninian Framework*. ACL93: Proc. of Annual Meeting of Association for Computational Linguistics. Association for Computational Linguistics, New Jersey, 1993a, pp. 105-111.

Kale, M.R. *A Higher Sanskrit Grammar*. 4th Ed,Motilal Banarasidass Publishers Pvt. Ltd.

Hopcroft, John E., Motwani, Rajeev, Ullman, Jeffrey D. 2002. *Introduction to Automata Theory, Languages and Computation*. 2^{nd} Ed, Pearson Education Pvt. Ltd., 2002.

SANSKRITTAGGER, A STOCHASTIC LEXICAL AND POS TAGGER FOR SANSKRIT

Oliver Hellwig

ABSTRACT

SanskritTagger is a stochastic tagger for unpreprocessed Sanskrit text. The tagger tokenises text with a Markov model and performs part-of-speech tagging with a Hidden Markov model. Parameters for these processes are estimated from a manually annotated corpus of currently about 1.500.000 words. The article sketches the tagging process, reports the results of tagging a few short passages of Sanskrit text and describes further improvements of the program.

The article describes design and function of SanskritTagger, a tokeniser and part-of-speech (POS) tagger, which analyses "natural", i.e. unannotated Sanskrit text by repeated application of stochastic models. This tagger has been developped during the last few years as part of a larger project for digitalisation of Sanskrit texts (cmp. (Hellwig, 2002)) and is still in the state of steady improvement. The article is organised as follows: Section 1 gives a short overview about linguistic problems found in Sanskrit texts which influenced the design of the tagger. Section 2 describes the actual implementation of the tagger. In section 3, the performance of the tagger is evaluated on short passages of text from different thematic areas. In addition, this section describes possible improvements in future versions.

1. INTRODUCTION

Concerning its analytical abilities, SanskritTagger is located quite at the bottom of a hierarchy of taggers. The tagger neither constructs a complete nor a partial syntactical analysis of a Sanskrit text. Instead, it only identifies the most probable lexical resolution for a given group of strings (tokenisation) and their most probable part-of-speech (POS) tags. In comparison with taggers for some European languages, this result might not seem very noteworthy. In fact, the limited abilities of this tagger are caused by the difficulties which Sanskrit poses to any tagging process especially during tokenisation and which are not encountered – at least in that degree – in the processing of European languages.

On a low phonological level, the euphonic rules called ***saṃdhi*** are certainly a serious obstacle to an easy tokenisation of Sanskrit text. While these regular phonological transformations can be resolved with automata (Huet, 2007) or by using a simple lookup strategy (see below, 2.2), they introduce a great deal of ambiguity in any analysis of Sanskrit text. Consider for example a long string where three points for *saṃdhi* splitting can be identified. Each of these *saṃdhi*s may be resolved in three different ways. Even in this simple example, $3 \cdot 3 \cdot 3 = 27$ new strings are generated by the complete resolution at the three splitting points.

The high number of candidate strings which must be checked for validity after *saṃdhi* resolution, leads directly to a group of connected phenomena which are in my opinion the central challenge for any automatic processing of Sanskrit: The extreme **morphological and lexical richness** of Sanskrit. Even if a moderately sized dictionary as in SanskritTagger is used, there exist about five million distinct inflected nominal and verbal forms which may be found in any text. (English, a language with a large vocabulary, has about one half of this number!) On one hand, opposite to languages as German and English, the rich morphology clarifies the functions of words in a phrase and therefore makes POS tagging (and parsing) easier. On the other hand, it is responsible for many analyses which are in fact just nonsensical (e.g. *āsane* ⇒ *ā* - *sane*, "to - in the gain"). These problems are aggravated by the peculiarities of Sanskrit lexicography.

The first important lexical phenomenon is the low **text coverage** of Sanskrit vocabulary. Compare the following figures for English texts (taken from (Waring and Nation, 1997)) and for the Sanskrit corpus which I collected during the last years:

Vocabulary size	Text coverage	
	English	Sanskrit
1000	72.0	60.8
2000	79.7	70.3
3000	84.0	75.2
4000	86.8	78.2
5000	88.7	80.3
6000	89.9	81.9

Although the English corpus is certainly better balanced than the Sanskrit corpus – meaning that texts from more diverse sources are included, what should actually lead to a decrease of text coverage –, the values for text coverage are clearly higher for English than for Sanskrit. Therefore, for tokenising even a simple Sanskrit text, a tagger must take into account a considerably higher number of lexemes than in other languages. This fact excludes to some extent ”easy solutions” as reduced vocabularies which proved useful in tagging (technical) texts in other languages.

Besides, Sanskrit has a great number of **homonyms**. A query in the program dictionary which took into account only homonymous words with the same grammatical category resulted in the following figures:[1]

nr. of homonyms	frequency
2	1949
3	112
more than 3	17

Among these homonyms, a lot of words with high frequency can be found as for example *kesara*, m. masc. with the four basic meanings ”mane”, ”(lotus) fibre”, ”(a plant name, prob.) Mesua ferrea L.” and (infrequent) ”name of a mountain” (but see LIPUR, 1, 72, 7 for a reference).

A further, often neglected difficulty is the almost total **lack of punctuation marks** in Sanskrit texts. Apart from *daṇḍa*s in narrative texts, which often mark the end of a (complex) narrative substructure, Sanskrit texts do not use any kind of reliable punctuation. *daṇḍa*s in metrical texts actually mark the end of a verse which often, but by far not regularly, coincides with the end of a syntactic structure as for example a subordinate clause. So, *daṇḍa*s may be helpful in generating hypotheses about the syntactic structure of a text, but can not be considered as punctuation marks in a strict sense. This lack has a far reaching effect on any tagging or parsing process applied to a Sanskrit text, because it can not be guaranteed that all words necessary for a complete analysis are really contained in the text delimited by these marks. Manual preprocessing of the text (e.g. insertion of a clear punctuation) can counteract this phenomenon, but certainly is contrary to the notion of natural, i.e. unpreprocessed text.

Finally, to understand Sanskrit texts correctly, it is often necessary to supplement a great deal of **implicit knowledge**. This situation occurs in two closely related areas. Firstly, texts which openly simulate speech acts as for example dialogues often necessitate the addition of central parts of speech as subject or objects. This phenomenon, which is also known from texts in other languages, is a still puzzling, but intensively studied topic in computational linguistics. Secondly, especially scientific texts in Sanskrit as commentaries or *sūtra*s frequently use a kind of prose which imitates an oral controversy between the proposers of differing opinions. Although this kind of prose is probably derived from real discussions, it uses a highly formalised language (for a description see e.g. (Hartmann, 1955)). ”Sentences” in this language frequently only offer few pieces of information which must be inserted in an implicit ”knowlegde frame” to be supplied by the reader. Consider for example the discussion of *sāpiṇḍya* in the PARĀŚARASMṚTIṬĪKĀ (on PARĀŚARADHARMASAṂHITĀ, *Ācārakāṇḍa*, 2, 15; (I.V. Vāmanaśarmā, 1893), 59). After the author has proposed the standard model of this kind of relation, which includes three generations into past and future starting from the *yajamāna*, an opponent objects that brother, uncle etc. of the *yajamāna* are not included in this model and therefore not related to the *yajamāna* by *sāpiṇḍya*. In the following reply of the author, information which is really supplied in the text is printed in bold characters:

[1]This is actually a quite strong constraint, as for example nouns of categories ”a masc.” and ”a neutr.”, which have the same unchangeable stem, differ only in few forms. Including these pseudo-homonymous words would increase the rate of homonyms up to ten percent of the total vocabulary.

> *maivam*
> This is **not like this**!
> *uddeśyadevataikyena kriyaikyasyātra vivakṣitatvāt*
> Brother, uncle etc. are included in this model because **the identity of the ritual is expressed by the identity of the gods invoked**.

I am not arguing that these phrases are not well formed. Nevertheless, their syntax and pragmatics can only be analysed correctly, after the pragmatics of the surrounding text has been analysed and "understood" by the computer. The same holds true for phenomena such as anaphora resolution. Actually, this is a task which in my opinion is by far too difficult for any automatical analysis of Sanskrit currently available.

2. IMPLEMENTATION OF THE TAGGER

To keep data and algorithms clearly separated, language specific information is stored in a *database*, while the *tagging routines* are implemented in `C++` with heavy use of `STL` classes. The following section describes these two central components of the tagging software.

2.1. The database

The first main component of the program, a relational database, which can be queried via `SQL`, stores dictionary, grammatical information, and a text corpus. The original dictionary was based on the digitalised version of Monier-Williams which was parsed with regular expressions to extract lexemes, meanings and grammatical categories. These information types were stored in separate tables in the database. During the last few years, the dictionary has been extended especially in the areas of *Āyurveda* and religious philosophy. It currently contains about 178.000 lexemes (172.000 nouns and 6.000 verbs) with about 185.000 associated grammatical categories and about 325.000 meanings.

An important issue in the processing of strongly inflectional languages as Sanskrit is the correct recognition of inflected forms. Here, I chose a twofold strategy. Inflected **nominal forms** are not stored in the database, but are recognised on the fly during the tagging process. For this task, all possible endings for any nominal grammatical category are stored in a separate table. During tagging, the last few letters of a given string are compared with these endings. If an ending matches the last letters of the string, the dictionary is searched for the first part of the string in the respective grammatical category. If a matching lexeme could be found in the dictionary, a new candidate is added to the set of possible solutions. Computationally, this approach speeds up the tagging process, because the lookup of the last few letters of a given string in an efficiently organised small set of endings is much less time consuming than a query from a database of over four millions inflected forms. Though the difference in duration only amounts to a few milliseconds per operation, the performance loss sums up to several seconds for a phrase of moderate size. Figure 1 sketches an example for this approach.

> *sanābhy**ām***
>
> > **ām** = acc. sg. of declension type **ā fem.**
> > dictionary lookup for (*sanābh*, **ā fem.**)
> > success ⇒ 1st candidate
>
> *sanābh**yām***
>
> > **yām** = loc. sg. of declension type **i adj.**
> > dictionary lookup for (*sanābh*, **i adj.**)
> > success ⇒ 2nd candidate
>
> . . .

Figure 1: Example for the analysis of nominal forms

On the contrary, inflected *verbal forms* are stored in the database. Currently, the database contains about 440.000 inflected verbal forms including forms derived from prefixed verbs. The decision to store the full verbal forms was not only motivated by the comparatively small number of forms, but also by the frequent irregularities and exceptions in the verbal system of Sanskrit. Of course, it is possible to construct automata which generate and accept correct verbal forms at runtime. Nevertheless, it seems to me that such an approach requires more effort than the design of a simple algorithm which generates correct verbal forms of the most typical grammatical classes and leaves the rest of the job (including correction of errors and input of rare or special forms) to the user. As in the case of nominal forms, the last few letters of possible verbs are checked before the database is queried.

Apart from the lexical and grammatical information,

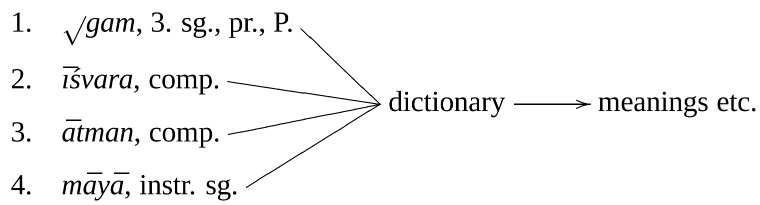


Figure 2: Analysis and storage of *gacchatīśvarātmamāyayā*

the database also stores a corpus of analysed Sanskrit texts which is of central importance to the tagging process. In short, every separable string of an input text is stored as a separate item in this corpus. After tagging this text, each of its strings is connected with an ordered list of references to grammatically annotated nouns from the dictionary and/or verbal forms. For example, the (fancy) string *gacchatīśvarātmamāyayā* is analysed and stored as sketched in figure 2. The figure makes clear the first important area of application of this corpus: Every string is resolved into lexemes or tokens, which are connected to the dictionary. The dictionary points in turn to further information about these lexemes as meanings etc. Of course, these relations may be inverted. Therefore, the corpus may be used to retrieve Sanskrit words efficiently in large amounts of text. Examples are the retrieval by lexeme ("Show all references in text X for the word Y!"), by meaning ("Show all references of words which have the meaning X!") or even by semantic concepts. Besides, the corpus is used to estimate the statistical parameters for the tagging process (see below). – In the moment, the corpus contains about 1.560.000 strings which are resolved into about 2.190.000 tokens. Each of these tokens is connected with a lexeme, and about 90 percent of them also have a POS annotation. Among others, the corpus includes the full analysed text of the RĀMĀYAṆA, the first books of the MAHĀBHĀRATA, some works of *dharma*- (e.g. MANUSMṚTI) and Purāṇa tradition, philosophical literature of Śaivism and many works on *Āyurveda* and *Rasaśāstra* (alchemy).

2.2. The tagging algorithm

The tagging software is divided into two main modules. In the first module, hypotheses about the analysis of a phrase are generated with the help of *saṃdhi* resolution and dictionary lookup. In the context of Sanskrit, a phrase does not mean a complete, self-contained syntactic structure, which may e.g. be extracted from a text with regular punctuation (see 1). Instead, a phrase $\mathcal{P}$ is a group of strings, i.e. words separated by blanks, which is terminated by a (double) *daṇḍa*. Such a group may, but needs not necessarily coincide with a complete syntactic structure. The hypotheses resulting from this first analysis are organised in an often complex tree- or rather forest-like structure. The purpose of the second module is to find the most probable lexical and morphological path through this structure given the statistical information extracted from the corpus. This path will constitute the final analysis of the phrase.

A string $\mathcal{S}_i \in \mathcal{P}$ of length L is parsed from left to right. At each position j with $1 \leq j \leq L$, a maximal number of n_{max} letters of the string are searched in a trie $\mathcal{T}$ whose nodes are sorted in binary order. $\mathcal{T}$ stores *saṃdhi* rules of the form $\mathcal{R} = \{s_{SRC}, s_1, s_2, type\}$ where s_{SRC} is the result of the *saṃdhi* between s_1 and s_2 and $type$ denotes the area of application of this *saṃdhi* (word, phrase, both). In trie terminology, s_{SRC} constitutes the path which leads to the leaves consisting of the reduced rules $\mathcal{R}' = \{s_1, s_2, type\}$. Obviously, multiple leaves may be assigned to a single path, as for example $\{\bar{a}, a, both\}$ and $\{a, a, both\}$ to the single letter path *ā*. n_{max} denotes the length of the longest s_{SRC} and is precalculated at program start.

If an extract s_{ijl} of $\mathcal{S}_i$ at position j matches a path s_{SRC} from the trie, l letters are removed from $\mathcal{S}_i$ beginning at position j. $\mathcal{S}_i$ is split into two new strings $\int_{i1}$ and $\int_{i2}$ at position j and the *saṃdhi* replacements s_1 and s_2 are affixed respectively prefixed to $\int_{i1}$ and $\int_{i2}$. Thereby, the new strings $\int_{i1} + s_1 = \mathcal{S}_{i1}$ and $s_2 + \int_{i2} = \mathcal{S}_{i2}$ are created. Figure 3 shows an example for this

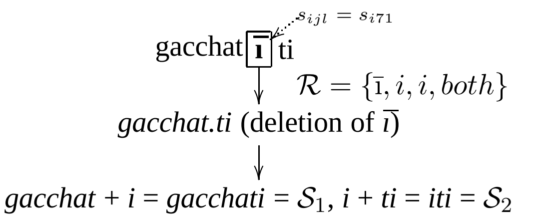


Figure 3: *saṃdhi* resolution for the string *gacchatīti*

approach.[2] To keep the *saṃdhi*-rule base simple, the program uses a recursive strategy for *saṃdhi* resolution. For example, a string $\mathcal{S}_{i1} =$ *xxxd* resulting from the rule $\mathcal{R} = \{dbh, d, bh, both\}$ can be transformed further by application of $\mathcal{R} = \{d, t, -, phrase\}$ into the form *xxxt*. After creation of $\mathcal{S}_{i1}$ and $\mathcal{S}_{i2}$, the *saṃdhi* routine is recursively called for these new strings which are treated in the same way as $\mathcal{S}_i$. If the running index j reaches the end of a string ($j = L$), it is checked if $\mathcal{S}_i$ is a valid Sanskrit form. The procedures for this check and the respective structures of the database were shortly described in section 2.1. If $\mathcal{S}_i$ is a valid Sanskrit form, the grammatical and lexical analysis of $\mathcal{S}_i$ are inserted into the analysis of $\mathcal{P}$.

During *saṃdhi* resolution and dictionary lookup, each $\mathcal{S}_i \in \mathcal{P}$ may have been resolved into m different subsets $\{\mathcal{S}_{i11}, \mathcal{S}_{i12}, \ldots, \mathcal{S}_{i1n_1}\}$, ..., $\{\mathcal{S}_{im1}, \ldots, \mathcal{S}_{imn_m}\}$ due to different *saṃdhi* resolution (SR). Furthermore, each of the substrings $\mathcal{S}_{ijk}$ has at least one grammatical and lexical analysis A_{ijkl} attached to it. Here, j is the number of the current SR, k the position of the substring in SR_j and l the index of the analysis for substring $\mathcal{S}_{ijk}$. Take for example the string $\mathcal{S}_1$ = *devadattakṛtakriyā*. (Parts of) two possible solutions (SR_1 and SR_2) are shown in figure 4 and 5. As indicated by the lines connecting the partial solutions A_{ijkl}, the second step of the tagging consists in finding the most probable path which runs through all $\mathcal{S}_i \in \mathcal{P}$. This step is again divided into two substeps. In the first substep, the most probable lexical path is searched with the help of a Markov model (MM). This path is fixed as the tokenisation of $\mathcal{P}$. In the second substep, the most probable syntactical analysis of this path is searched with a HMM.

[2]The expression s_{i71} in figure 3 is not a mistake. Because *ch* is treated as a single phoneme, it is replaced with a single letter in the internal representation.

The first substep, i.e. the **tokenisation** can be modelled with a discrete, first-order Markov chain (see e.g. (Rabiner, 1989) for a readable introduction). The MM is based on the concept of conditional probability, which is the probability of event B given the occurrence of another event A: $P(B \mid A) = \frac{P(A \cap B)}{P(A)}$. In the given context, if $x_{[a,b]}$ denotes an ordered sequence of lexemes with decreasing indices i ($a \geq i \geq b$), the probability that phrase $\mathcal{P}$ of length L is tokenised into lexemes $x_1, x_2, \ldots x_L$ is given as

$$\begin{aligned} \underbrace{p(x)}_{P(A\cap B)} &= \underbrace{p(x_L \mid x_{[L-1,1]})}_{P(B|A)} \cdot \underbrace{p(x_{[L-1,1]})}_{P(A)} \\ &= p(x_L \mid x_{[L-1,1]}) \cdot p(x_{L-1} \mid x_{[L-2,1]}) \\ &\quad \cdot p(x_{[L-2,1]}) \quad \text{etc.} \end{aligned} \tag{1}$$

Under the assumption, that the probability of each lexeme depends only on its direct predecessor (first-order model), the formula is simplified to

$$\begin{aligned} p(x) &= p(x_L \mid x_{L-1}) \cdot p(x_{L-1} \mid x_{L-2}) \cdot \ldots \cdot p(x_1) \\ &= p(x_1)\Pi_{i=2}^{L} p(x_i \mid x_{i-1}) \end{aligned} \tag{2}$$

For reasons of floating point accuracy, equation 2 is transformed into

$$p(x) = \log p(x_1) + \sum_{i=2}^{L} \log p(x_i \mid x_{i-1}) \tag{3}$$

with $\log(a \cdot b) = \log a + \log b$. To find the most probable path, a modified form of the well known *Viterbi algorithm* is used. This algorithm was actually designed for HMMs, but can be applied to the given problem due to its similar structure. Before applying this algorithm to the data, it should be taken into consideration, that any string $\mathcal{S}_i$ may have been resolved in subsets S_{ix} and S_{iy} with different sizes $|S_{ix}| \neq |S_{iy}|$. Therefore, the algorithm can not be applied naively to the hypotheses generated in the first step. Instead, for each Phrase $\mathcal{P}$ which contains N strings $\mathcal{S}_1, \ldots, \mathcal{S}_N$, a vector of length N is allocated which stores for every string $\mathcal{S}_i$ the currently checked index j of its *saṃdhi* resolutions. So, a vector beginning in $\boxed{1}\boxed{2}\boxed{1}\ldots$ means, that currently the first resolution of $\mathcal{S}_1$, the second resolution of $\mathcal{S}_2$ and the first resolution of $\mathcal{S}_3$ are

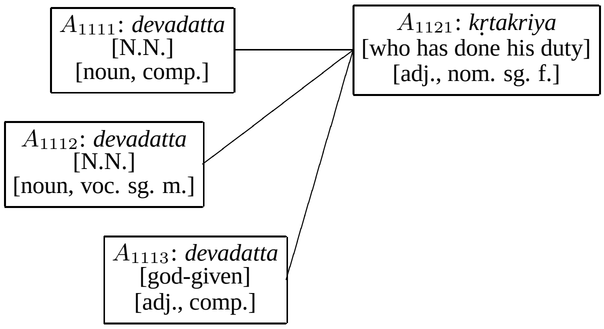


Figure 4: A possible analysis of the string *devadattakṛtakriyā*

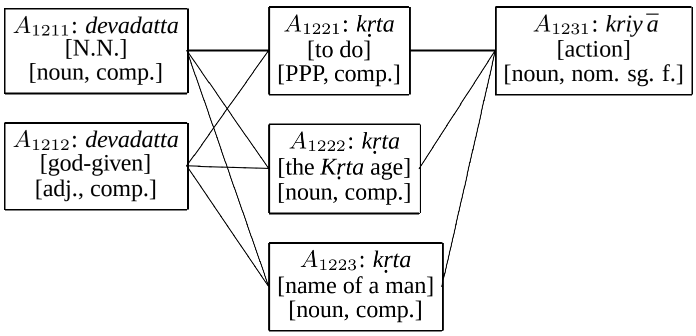


Figure 5: Another possible analysis of the string *devadattakṛtakriyā*

checked. Such a combination of the analyses of successive strings will be called *path subset* (PS). (Note that the diagrams which show possible resolutions of *devadattakṛtakriyā* each constitute one path subset!) If n_i denotes the number of SRs for $\mathcal{S}_i$, there exist $\Pi_{i=1}^{N} n_i$ different PSs. Now, among all PSs, the PS with the best path regarding lexical probability is searched with a modified version of the Viterbi algorithm. Obviously, some PS consist of shorter paths because the strings in these combinations were split at fewer *saṃdhi* points. To treat all PS equally, they are filled up with "dummy probabilities" which are calculated as the mean of the probabilities constituting the best path $\pi_{opt\,u} \in PS_u$. If l_{max} denotes the length of the longest PS, l_u the length of the current PS_u and $\bar{p}(\pi_{opt\,u})$ the average transition probability between all elements constituting the optimal path in PS_u, the value $\sum_{r=l_{max}-l_u}^{l_{max}} \log \bar{p}(\pi_{opt\,u})$ is added to the probability of $\pi_{opt\,u}$.

The most probable path found with this algorithm is considered as the tokenisation of $\mathcal{P}$, which is then annotated morphologically with the help of an HMM. The POS tagset used for this annotation contains the 136 items shown in table 1. To understand the relation between this tagset and the actual morphological analysis of a word, consider the string *gacchati*. After the program has fixed the lexeme *gam* ("to go") as its most probable lexical analysis, there exist three possible morphological resolutions: "he/she/it goes" (3rd, sg., pres., P.) and "in the going ..." (loc. sg., masc./neutr., part. pres., P.). The solution "he ... goes" is mapped to the POS tag ["present tenses", 3rd, sg.], while the nominal solutions are mapped to ["present participles", loc., sg., masc.] and ["present participles", loc., sg., neutr.], respectively. Note, that a good deal of information is lost during this mapping process. For example, no distinction is made between different present tenses. Instead, forms like *gacchati* and *gacchatu* are considered to be syntactically equivalent and are therefore mapped to the same POS tag. This loss of information reflects the decision between the granularity of the tagset and the amount of text from which the probabilities of the tags can be estimated – the more text is available, the finer a granularity can be chosen.

The HMM $\lambda = \{A, B, \pi\}$ used to simulate the syntactical structure of $\mathcal{P}$ consists of the probability distribution A of transitions between tags (= states), the observation symbol probability distribution B, which records the probabilities $p(x \mid T)$ that a lexeme x is emitted given the tag T, and the initial state distribution π, i.e. the probabilities with which a tag opens a phrase (cmp. (Rabiner, 1989), 260/61). The values in A can be estimated from the corpus. The probabilities in B can be calculated using *Bayes theorem* $p(T) \cdot p(x \mid T) = p(x) \cdot p(T \mid x)$, where $p(x)$, $p(T)$ and $p(T \mid x)$ again can be estimated from the corpus. As indicated above, the optimal path with regard to POS probability is again searched with the Viterbi algorithm. This path is finally presented to the user as the most probable resolution of a phrase, which may be accepted and stored in the database or (manually) replaced with a better solution.

3. PERFORMANCE AND IMPROVEMENTS

This section gives the results of tagging some short passages of text. The results are in no way representative. They are only meant to demonstrate the performance of the algorithm on different types of Sanskrit text. Three types of errors are distinguished. A saṃdhi *error* (e_S) occurs if a string is split at wrong splitting points. This error invalidates the results for the whole string. A *lexical error* (e_L) indicates that a string was split at correct *saṃdhi* points, but that a wrong lexeme was activated during tokenisation. Finally, a *POS error* (e_{POS}) occurs if a wrong POS tag was assigned to a correct token. Furthermore, the value r_{SL} gives the ratio of strings to lexemes, i.e. $r_{SL} = \frac{\text{nr. of strings}}{\text{nr. of lexemes}}$. A high value of r_{SL} indicates that the passage uses few composite words. – The following five passages were analysed:

1. LIṄGAPURĀṆA, 2, 20, 1-10: A *Purāṇic* text which treats a Śivaite topic. Easy verses.

2. VIṢṆUSMṚTI, 63, 35-50: An example of the scientific style. Many supplements are needed to get the full meaning of the passage.

3. MŪLAMADHYAMAKĀRIKĀ of Nāgārjuna, 12, 1-10: Easy Buddhist prose (from a linguistic point of view!).

4. GĪTAGOVINDA of , 1.2-5: Poetry with many unusual words.

Verbal forms		
Finite verbal forms		
present tenses (incl. imperative and opt.)	×9: person, number	9
past tenses	×9: person, number	9
future tenses	×9: person, number	9
other tenses	×9: person, number	9
Infinite verbal forms		
absolutive		1
infinitive		1
past participle, gerund	×24: case, nr., gender	24
present participles	×24: case, nr., gender	24
other participles	×24: case, nr., gender	24
Nominal forms		
indeclinable		1
nouns in composite words		1
nouns, adjectives	×24: case, nr., gender	24
		136

Table 1: Tagset used for POS tagging of Sanskrit text

5. KĀMASŪTRA, 2, 1, 1-12: Scientific prose.

The results, which are displayed in table 2, support the assumptions about the problems which are encountered in tagging natural Sanskrit text (see section 1). The tagger performs best on texts which are written in an easy style and come from "well known" areas of knowlegde (1, 3). On the contrary, a difficult vocabulary (5) and demanding syntactical structures (4) introduce a great deal of *saṃdhi* (4, 5) and POS (5) errors. The comparatively high number of POS errors in 3 is above all caused by confusion between nom. and acc. sg. neutre and could certainly be reduced by training the tagger with only a few similar texts.

At the moment, three main areas for **improvement** of this tagger can be discerned. Firstly, a reliable **estimation of probability values** for rare lexemes and for infrequent POS combinations is the central step in improving the tagger. Application of the classical forward-backward-reestimation actually lead to degradation of the probability values (cmp. (Abney, 1996), 3). Although many other methods such as smoothing probabilities or the use of neural networks were proposed, Sanskrit offers a (partial) solution of this problem which is totally based on its lexicography. Sanskrit not only possesses a high number of homonymous, but also of synonymous words. Many of these words are already integrated in a semantic network (based on the `OpenCyc` ontology), which is contained in the program database. To estimate probabilities, groups of synonyms can be identified which designate the same sememe with a high degree of probability. In this sense, the group "horse" is constituted by {*aśva*, *turaga*, *turaṃga*, *turaṃgama*, *vājin*, *haya*}, but not *hari*, which means "Viṣṇu" in most cases. If one of the words which is included in the group "horse" is met in an unknown context (either lexical or POS/morphological), the respective probabilities can be estimated from the values given for other members of the group.

Secondly, **integration of rules** can certainly improve analysis. Due to lack of punctuation (see 1), these rules should not describe well-formed and complete phrases, but only check the coherence of few members of a phrase $\mathcal{P}$, i.e. a syntactic substructure delimited by *daṇḍa*s (chunk parsing). Some preliminary tests with rules which reject paths during POS tagging on the base of simple syntactic criteria turned out to be successful.

Thirdly and finally, the strict **separation of tokenisation and POS tagging** is a constant source of errors. Consider, for example, the simple sentence *brahmā varaṃ te dāsyati* ("Brahmā will give you a boon.").

nr.	P	S	L	r_{SL}	e_S	e_L	e_{POS}	corr. phrases
1	22	89	139	0.64	4	6	2	10
2	17	48	88	0.55	2	4	6	5
3	20	135	157	0.86	0	2	9	13
4	8	49	86	0.57	8	4	1	0
5	24	123	167	0.74	8	5	15	10

Table 2: Error rates of the tagger in five short passages – Abbreviations: **nr.**: number of the passage, **P**: number of phrases, **S**: number of strings, **L**: number of lexemes

Although the POS sequence [dat. sg.] - [3. sg. fut.] is well established and would always be preferred to the incongruent [nom. pl.] - [3. sg. fut.], during tokenisation *te* is interpreted as the nom. pl. masc. of the pronoun *tad* due to its enormous frequency. Because the content of the best lexical path can not be changed during POS analysis, the correct analysis for *te* (dat. sg. of *tvad*) will never be activated. Possible workarounds for this problem are a more flexible POS analysis, which takes into account e.g. the first five lexical resolutions, or a combination of tokenisation and POS tagging in one procedure.

4. REFERENCES

1893. *Parāśaradharmasaṃhitā. Ācārakāṇḍam*.

Hartmann, Peter. 1955. *Nominale Ausdrucksformen im wissenschaftlichen Sanskrit*. Carl Winter Universitätsverlag, Heidelberg.

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286.

Steven Abney. 1996. In *Corpus-Based Methods in Language and Speech*, Dordrecht. Kluwer Academic Publishers.

Robert Waring and Paul Nation. In *Vocabulary: Description, Acquisition and Pedagogy*. pages, 6–19, Cambridge. Cambridge University Press.

Hellwig, Oliver. 2002. *Sanskrit und Computer*. Ph.D. thesis, Freie Universit̄at Berlin.

Gérard Huet. 2007. Shallow syntax analysis in sanskrit guided by semantic nets constraints. In *International Workshop on Research Issues in Digital Libraries*.

Inflectional Morphology Analyzer for Sanskrit

Girish Nath Jha, Muktanand Agrawal, Subash, Sudhir K. Mishra, Diwakar Mani, Diwakar Mishra, Manji Bhadra, Surjit K. Singh

girishj@mail.jnu.ac.in

Special Centre for Sanskrit Studies
Jawaharlal Nehru University
New Delhi-110067

The paper describes a Sanskrit morphological analyzer that identifies and analyzes inflected noun-forms and verb-forms in any given sandhi free text. The system which has been developed as java servlet RDBMS can be tested at http://sanskrit.jnu.ac.in (Language Processing Tools > Sanskrit Tinanta Analyzer/Subanta Analyzer) with Sanskrit data in Unicode text. Subsequently, the separate systems of subanta and tiṅanta will be combined into a single system of sentence analysis with karaka interpretation. Currently, the system checks and labels each word as three basic POS categories - subanta, tiṅanta, and avyaya. Thereafter, each subanta is sent for subanta processing based on an example database and a rule database. The verbs are examined based on a database of verb roots and forms as well by reverse morphology based on Paninian techniques. Future enhancements include plugging in the amarakosha (http://sanskrit.jnu.ac.in/amara) and other noun lexicons with the subanta system. The tiṅanta will be enhanced by the krdanta analysis module being developed separately.

1. Introduction

The authors in the present paper are describing the *subanta* and *tiṅanta* analysis systems for Sanskrit which are currently running at http://sanskrit.jnu.ac.in. Sanskrit is a heavily inflected language, and depends on nominal and verbal inflections for communication of meaning. A fully inflected unit is called *pada*. The *subanta padas* are the inflected nouns and the *tiṅanta padas* are the inflected verbs. Hence identifying and analyzing these inflections are critical to any further processing of Sanskrit.

The results from the *subanta* analyzer for the input text fragment

आम्रस्य आत्मकथा
चपलाः बालकाः आम्राणाम् उद्यानं गच्छन्ति । तत्र आम्रफलानि पश्यन्ति प्रसन्नाः च भवन्ति ।

are displayed as follows –

आम्रस्य_SUBANTA_SUBANTA आत्मकथा [आत्मकथा (स्त्रीलिङ्ग) + सु, प्रथमा, एकवचन] [*_PUNCT]
चपलाः [चपल (पुल्लिङ्ग) + जस्, प्रथमा, बहुवचन] बालकाः [बालक (पुल्लिङ्ग) + जस्, प्रथमा, बहुवचन]
आम्राणाम् [आम्र (पुल्लिङ्ग) + आम्, षष्ठी, बहुवचन] उद्यानं [उद्यन्+अम्, द्वितीया, एकवचन]
[गच्छन्ति_VERB] [।_PUNCT] [तत्र_AV] आम्रफलानि [आम्रफल+जस्/शस् प्रथमा/द्वितीया, बहुवचन]
[पश्यन्ति_VERB] प्रसन्नाः [प्रसन्न (पुल्लिङ्ग) + जस्, प्रथमा, बहुवचन] [च_AV] [भवन्ति_VERB]
[।_PUNCT]

Those colored blue are non *subanta* categories and those colored red are possible errors. The default black colored ones are the *subanta* padas analyzed.

The word गच्छन्ति from the above input text resulted in the following output from the *tiṅanta* analyzer system –

गच्छन्ति { (कर्तृवाच्य) गम् ([भ्वादिगण] [अनिट्] [सकर्मक]) ([लट्]) झि ([परस्मै] [
प्रथम–पुरुष] [बहुवचन]) }

2. The *Subanta* Anlyzer

The system accepts Unicode (UTF-8) sandhi free Devanagari Sanskrit inputs (word, sentence or text) and processes it according to the following sequence -

INPUT TEXT
↓
PRE-PROCESSOR
↓
VERB DATABASE→ LIGHT POS TAGGING ←AVYAYA DATABASE
↓
SUBANTA RECOGNIZER←VIBHAKTI DATABASE
↓
SUBANTA RULES→ *SUBANTA ANALYZER* ←SANDHI RULES
↓
SUBANTA ANALYSIS

The PREPROCESSOR does the simplification and normalization of the Sanskrit text (for example, deletes Roman characters, other invalid words, punctuations etc). The POS TAGGER identifies four categories AVyaya, VERB, PUNCTuation and SUBANTA. The SUBANTA RECOGNIZER does *vibhakti* identification and isolation by searching the *vibhakti* database. The SUBANTA ANALYZER does analysis by checking the *subanta* rule base and sandhi rules. Analysis includes splitting the NPs into its constituents - base [(*prātipadika*) (PDK)], case-number markers (*kāraka-vacana-vibhakti*).

3. Sanskrit sentence and basic POS categories

A Sanskrit sentence has NPs (including AVs), and VPs. Cordona[1] (1988) defines a sentence as -

$$(N – E^n)\ p \ldots (V – E^v)\ p$$

After *sup* and *tiṅ* combine with PDK, they are assigned syntactico-semantic relation by the *kāraka* stipulations to return complete sentences.

---

[1] George Cardona, 1988 Pāṇini, His Work and its Traditions, vol **...** i (Delhi: MLBD, 1988)

3.1 Sanskrit *subanta* (inflected nouns)

Sanskrit nouns are inflected with seven case markers in three numbers. Potentially, a noun can be declined in all three genders. Sanskrit noun forms can be further complicated by being a derived noun as primary (*kṛdanta*), secondary (*taddhitānta*), feminine forms (*strīpratyayānta)* and compounds (*samāsa).* They can also include *upasargas* and AVs etc. According to Pāṇini, there are 21 case suffixes called *sup* (seven *vibhaktis* combined with three numbers)[2], which can attach to the nominal bases (PDK) according to the syntactic category, gender and end-character of the base. Pāṇini has listed these as sets of three as:

su, au, jas
am, auṭ, śas
ṭā, bhyām, bhis
ṅe, bhyām, bhyas
ṅasi, bhyām, bhyas
ṅas, os, ām
ṅi, os, sup[3]

for singular, dual and plural[4] respectively. These suffixes are added to the PDKs[5] (any meaningful form of a word, which is neither a root nor a suffix) to obtain inflected forms NPs. PDKs are of two types: primitive and derived. The primitive bases are stored in *gaṇapāṭha* [(GP) (collection of bases with similar forms)] while the latter are formed by adding the derivational suffixes. NPs are of mainly six types –

3.1.1 *avyaya subanta* (indeclinable nouns)

Avyaya subanta, remain unchanged under all morphological conditions[6]. According to Pāṇini [2.2.82][7], affixes *cāp, ṭāp, ḍāp,* (feminine suffixes) and *sup* are deleted by *luk* when they occur after an AVs. Pāṇini defines AVs as *svarādinipātamavyayam* [1.1.36]*, kṛnmejantaḥ* [1.1.38], *ktvā tosun kasunaḥ* [1.139] and *avyayībhāvaśca* [1.1.40][8] etc.

3.1.2 basic *subantas* (primitive nouns)

Basic *subantas* are formed by primitive PDKs found in the Panini's *gaṇapāṭha*. For our purpose, all those nouns, the base or inflected form of which can be found in a lexicon can be considered basic *subantas*. Sometimes, commonly occurring primary or secondary derived nouns, feminine or compound forms can also be found in the lexicon. Therefore such *subantas* are also considered basic and do not require any reverse derivational analysis unless specifically required.

---

2 स्वौजसमौट्छष्टाभ्याम्भिस्ङेभ्याम्भ्यस्ङसिभ्याम्भ्यस्ङसोसांङ्योस्सुप्

3 सुपः

4 द्व्येकयोर्द्विवचनैकवचने

5 अर्थवधातुरप्रत्ययः प्रातिपदिकम् ।१।२।४५।।, कृत्तद्धितसमासाश्च ।१।२।४६।।

6 सदृशं त्रिषु लिङ्गेषु सर्वासु च विभक्तिषु ।
वचनेषु च सर्वेषु यन्न व्येति तदव्ययम् ।। [गोपथ ब्राह्मण]

7 अव्ययादाप्सुपः [२.४.८२]

8 स्वरादिनिपातमव्ययम् [१.१.३६], कृन्मेजन्तः [१.१.३८], क्त्वा–तोसुन्–कसुनः [१.१.३९], अव्ययीभावश्च [१.१.४०]

Such inflected nouns are formed by inflecting the base or PDKs (*arthavadadhāturapratyayaḥ prātipadiakam)* with *sup*. For example: *rāmaḥ, śyāmaḥ, pustakālayaḥ, vidyālayaḥ* etc.

3.1.3 *samāsānta subanta* (compound nouns)

Simple words (*padas*), whether substantives, adjectives, verbs or indeclinables, when added with other nouns, form *samāsa* (compound). Sanskrit *samāsas* are divided into four categories, some of which are divided into sub-categories. The four main categories of compounds are as follows:

- adverbial or *avyayībhāva*,
- determinative or *tatpuruṣa,*
- attributive or *bahuvrīhi* and
- copulative or *dvandva*. *dvandva* and *tatpuruṣa* compounds may be further divided into sub-categories

3.1.4 *kṛdanta subanta* (primary derived nouns)

The primary affixes called *kṛt* are added to verbs to derive substantives, adjectives or indeclinables.

3.1.5 *taddhitānta subanta* (secondary derived nouns)

The secondary derivative affixes called *taddhita* derive secondary nouns from primary nouns. For example - *dāśarathī, gauṇa* etc.

3.1.6 *strīpratyayānta subanta* (feminine derived nouns)

Sanskrit has eight feminine suffixes *ṭāp, cāp ḍāp, ṅīṣ, ṅīn, ṅīp, uṅ* and *ti* etc. and the words ending in these suffixes are called *strīpratyayānta* For example - *ajā, gaurī, mūṣikā, indrāṇī, gopī, aṣṭādhyāyī, kurucarī, yuvatī, karabhorū* etc.

4. Recognition of Sanskrit *subanta*

4.1 Recognition of punctuations

System recognizes punctuations and tags them with the label PUNCT. If the input has any extraneous characters, then the input word will be cleaned from these elements (i.e. ‘normalized’) so that only Devanāgarī Sanskrit input text is sent to the analyzer. For example, “रा/&^%@#म:, बा,’”:–=लक:” ➔ राम:, बालक:

4.2 Recognition of *Avyayas*

System takes the help of *avyaya* database for recognizing AVs. If an input word is found in the AVs database, it is labeled AV, and excluded from the *subanta* analysis as AVs do not change forms after *subanta* affixation. We have stored most AVs in the *avyaya* database.

4.3 Recognition of verbs

System takes the help of verb database for verb recognition. If an input is found in the verb database, it is labeled VERB and thus excluded from *subanta* analysis. Since storing all Sanskrit verb forms is not possible, we have stored verb forms of commonly used 450 verb roots.

4.4 Recognition of *subanta*

Thus, a process of exclusion identifies the nouns in a Sanskrit text. After the punctuations, *avyayas* and verbs are identified, the remaining words in the text are labeled SUBANTA.

5. Analysis of *subanta*

System does analysis of inflected nouns with the help of two relational database - examples and rules. Brief description of these databases follows-

5.1 Example database

All complicated forms (which are not analyzed according to any rule) including those of some pronoun are stored the database. For example: अहम्=अस्मद+सु प्रथमा एकवचन;अहं=अस्मद+सु प्रथमा एकवचन;आवाम्=अस्मद+औ प्रथमा द्वितीया द्विवचन;आवां=अस्मद+औ प्रथमा द्वितीया द्विवचन;वयम्=अस्मद+जस प्रथमा बहुवचन;वयं=अस्मद+जस प्रथमा बहुवचन;माम्=अस्मद+अम द्वितीया एकवचन;मां=अस्मद+अम द्वितीया एकवचन

5.2 Rule database

The *subanta* patterns are stored in this database. This database analyzes those nouns which match a particular pattern from the rule base. For example, रामः, नदी, रमा, पुस्तकम etc. First, the system recognizes *vibhakti* as the end character of nouns. For example, '**:**' is found in nominative singular (1-1) like -रामः, श्यामः, सर्वः, भरतः एकः . The system isolates '**:**' and searches for analysis in the *sup* rule base. In the case of nominative and accusative dual (1-2/2-2), PDK forms will be 'ौ' ending, for example - रामौ, श्यामौ, सर्वौ, एकौ. The system isolates 'ौ' and searches for analysis by matching in the rule database. The sample data is as follows –

ा=ा+सु प्रथमा एकवचन;ाभ्याम्=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;ाभ्यां=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;भ्याम्=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;भ्यां=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;ेभ्यः=+भ्यस् चतुर्थी पञ्चमी बहुवचन;भ्यः=+भ्यस् चतुर्थी पञ्चमी बहुवचन;

5.3 verb data sample

भवति,भवतः,भवन्ति,भवसि,भवथः,भवथ,भवामि,भवावः,भवामः,भवतु,भवताम्,भवन्तु,भव,भवतम्,भवत,भवानि,भवाव,
भवाम,अभवत्,अभवताम्,अभवन्,अभवः,अभवतम्,अभवत,अभवम्,अभवाव,अभवाम,भवेत्,भवेताम्,भवेयुः,भवेः,भ
ोतम्,भवेत,भवेयम्,भवेव,भवेम,बभूव,बभूवतुः,बभूवुः,बभूविथ,बभूवथुः,बभूव,बभूव,बभूविव

5.4 *avyaya* data sample

अ,कश्चित्,सदैव,अकस्मात्,अकाण्डे,अग्निसात्,अग्नी,अघोः,अङ्ग,अजस्रम्,अञ्जसा,अतः,अति,अतीव,अत्र,अथ,अथ
ाकिम्,अथवा,अथो,अद्धा,अद्य,अद्यापि,अधरात्,अधरेद्युः,अधरेण,अधः,अधस्तात्,अधि,अधिहरि,अधुना,अधोऽधः,अध्यय
ानतः,अनिशम्,अनु,अनेकधा,अनेकशः,अन्तः,अन्तरा,अन्तरेण,अन्यतः,अन्यत्,अन्यत्र

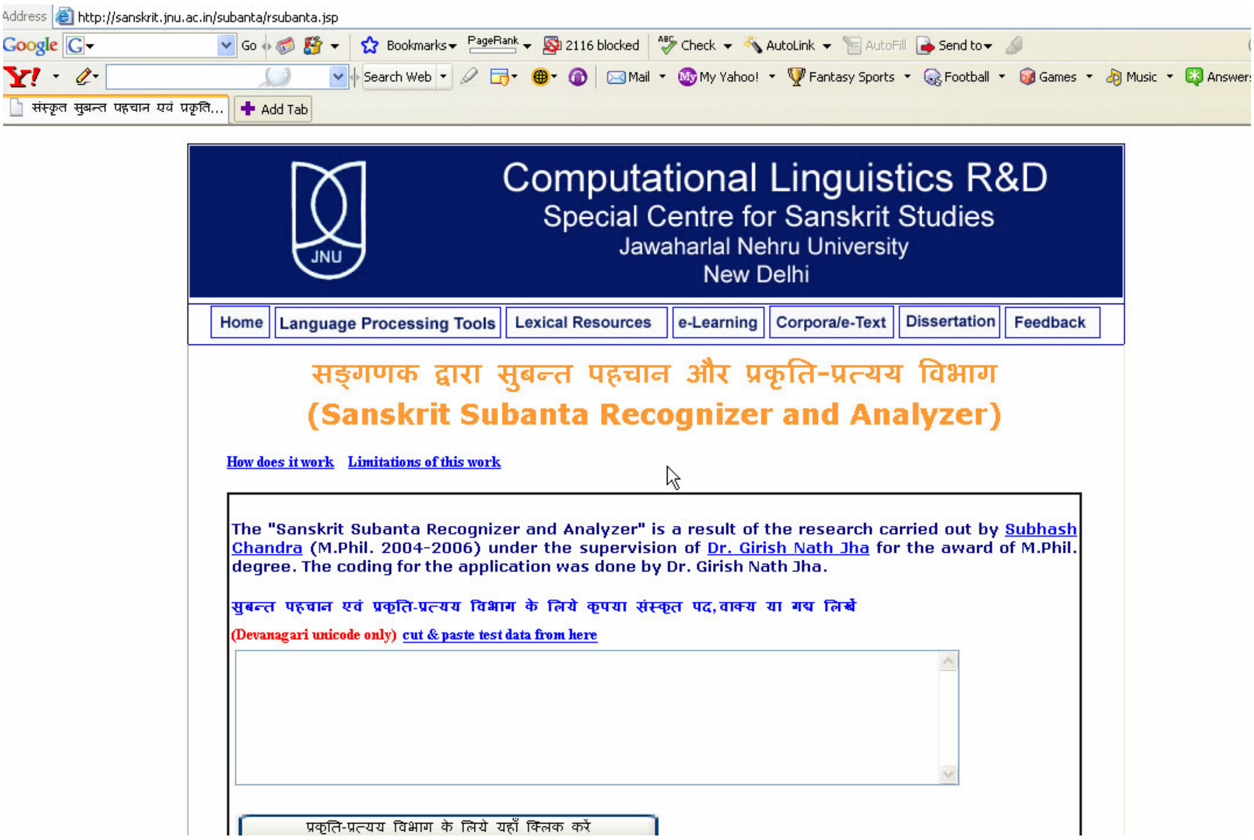
5.5 Architecture of the system

The following model describes the interaction between multi-tiered architecture of the *subanta* analyzer:

U S E R
↓ ↑
Request **response**
↓ ↑
Apache-tomcat
↓↑
Java servlet
↓↑
JDBC
↓↑
Database

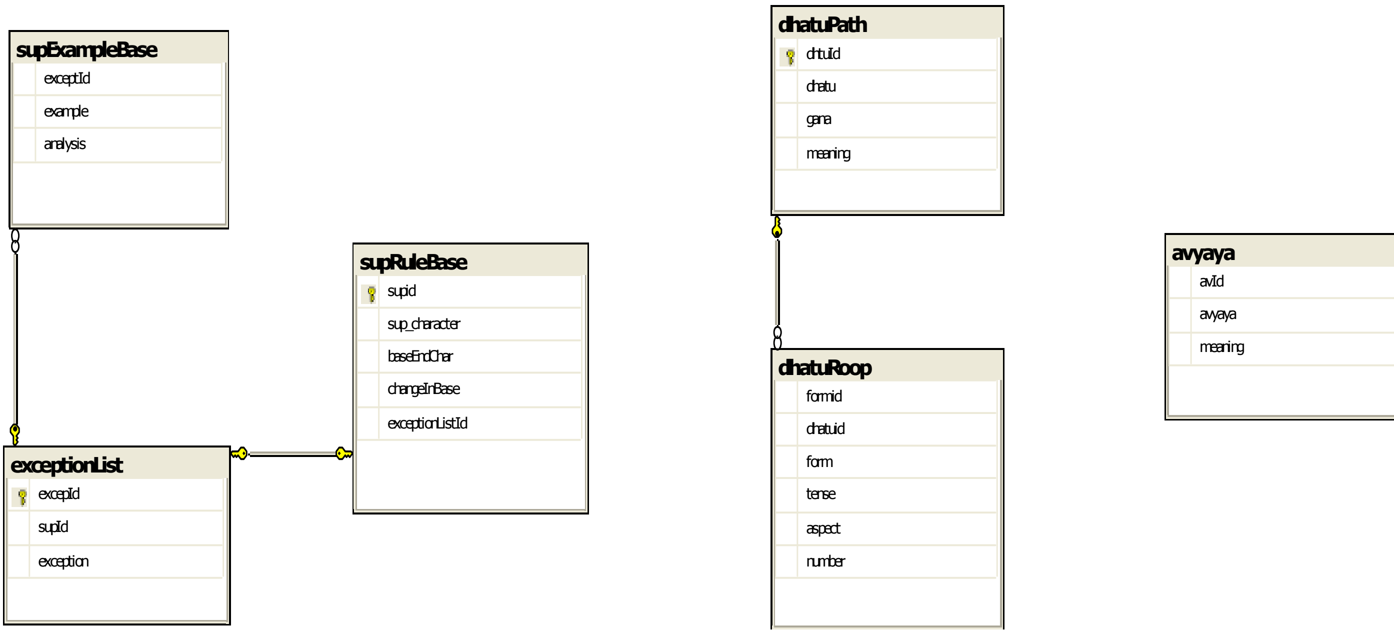
5.6 Front-end: online interface

The Graphical User Interface (GUI) is produced by JSP (Java Server Pages). The JSP interface allows the user to give input in Devanagari utf-8 format using HTML text area component. The user interface is displayed as follows:



5.7 Back-end: database / txt files

There are two versions of the system; the server-based version connects to a MSSQL Server 2005 RDBMS through JDBC. The rule base, example base and other linguistic resources are stored as Devanagari utf-8. The PC based portable version, for obvious reasons, cannot have RDBMS support. Therefore, we have our rules and data stored in utf-8 text files as backend. A design of the reverse *subanta* database is given below-



The *supRuleBase* table has relations with the *exceptionList* table. Any exception figuring in the rule base must have a description in the exception list. The table *supExampleBase* depends on the *exceptionList* and must provide analysis for each example figuring in the *exceptionList* and marked in the *supRuleBase.* The *dhāturūpa* object depends on the *dhātupāṭha* object while the AVs is a floating object as of now. These linguistic resources are checked for recognition of nouns, and the rules and example bases are searched for analysis. System uses some text/data files whose samples have been given in earlier sections.

5.2 Database connectivity

The database connectivity is done through Java Database connectivity (JDBC) driver. JDBC Application Programming Interface (API) is the industry standard for database independent connectivity for Java and a wide range of SQL databases. JDBC technology allows using the Java programming language to develop 'Write once, run anywhere' capabilities for applications that require access to large-scale data. JDBC works as bridge between Java program and Database. SQL server 2005 and JDBC support input and output in Unicode, so this system accepts Unicode Devanagari text as well as prints result in Unicode Devanagri too[9].

6. Limitations of the system

6.1 Limitations of the recognition process

This system has the following recognition limitations:

- at present, we have approximately verb forms for only 450 commonly found verb roots in the verb database. Though it is very unlikely that ordinary Sanskrit literature will overshoot this list, yet the system is likely to start processing verb forms as nouns if not found in this limited database.
- at this point, the system will wrongly mark prefixed or derived verb-forms as nouns as they will not be found in the verb database. The gains from the *tiṅanta* analyzer will be added here shortly to overcome this limitation.
- currently this work assumes sandhi free text. So, a noun or verb with sandhi is likely to return wrong results. The gains from a separate research on sandhi processing will be used to minimize such errors.
- currently, our AV database has only 519 AVs. It is not enough for AV recognition in ordinary Sanskrit literature. In this case, the system is likely to start processing AVs as nouns, if it is not found in AVs database.
- some forms ending in primary affixes look like nouns while they are AVs. For example: पठितुम्, गत्त्वा, आदाय, विहस्य etc. System will incorrectly recognize and process them as *subantas*.
- many nouns (for example, *śtṛ pratyayānta* in locative singular) look like verbs. These will be wrongly recognized as verbs for example: भवति, गच्छति, पठति, चलति etc. To solve this problem, we will have a hybrid POS category called SUPTIN for those verb forms which are *subantas* as well.

[9] http://java.sun.com/products/servlet/

6.2 Limitations of the analysis process

The system has the following analysis limitations:

- same forms are available in the dual of nominative and accusative cases, for example, रामौ, dual of instrumental, dative and ablative cases, for example रामाभ्याम्, plural of dative and ablative cases, for example रामेभ्यः, dual of genitive and locative cases, for example रामयोः. In neuter gender as well, the nominative and accusative singular forms may be identical as in पुस्तकम् (1-1 and 2-1). In such cases, the system will give all possible results as in

रामौ	=	औ	[प्र./ द्वि. द्विव.]
रामाभ्याम्	=	भ्याम्	[तृ./च./पं. द्विव.]
रामेभ्यः	=	भ्यस्	[च./पं. बहुव.]
रामयोः	=	ओस्	[ष./स. द्विव.]
पुस्तकम्	=	सु/अम्	[प्र./द्वि. एकव.]
हरेः	=	ङसि/ङस्	[पं./ष. एकव.]

- some *kṛdanta* forms (generally *lyap, tumun,* and *ktvā* suffix ending) look like nouns (for example - विहस्य पठित्वा, गत्वा, पठितुम्, गन्तुम, नेतुम्, प्रदाय, विहाय etc.). In such cases, the system may give wrong results as:

  विहस्य = विह + ङस् षष्ठी एकवचन
  पठित्वा = पठित्वा + सु प्रथमा एकवचन
  गत्वा = गत्वा + सु प्रथमा एकवचन
  पठितुम् = पठितु + अम् द्वितीया एकवचन
  गन्तुम् = गन्तु + अम् द्वितीया एकवचन

- at this point, system does not have gender information for all PDKs, nor does it attempt to guess the gender. This limitation is going to be minimized by plugging in the *amarakośa* shortly.

- currently this system is giving multiple results in ambiguous cases, because the words as analyzed a single tokens. This will be solved by adding the gains from the research on *kāraka* and gender of nouns which concluded recently.

7. The *tiṅanta* analyzer

Verbs constitute an important part of any language. A sentence indispensably requires a verb to convey complete sense. Given the importance of verb and verb phrases in any linguistic data, it is necessary to develop a proper strategy to analyze them. Creating lexical resource for verbs along with other parts of speech is a necessary requirement. Sanskrit is a highly inflectional language. It is relatively free word-order language. The semantic inter-relation among the various components of a sentence is established through the inflectional suffixes.

Scholars have done efforts to analyze Sanskrit verb morphology, both in theory and in computation. Some of the major works are listed below:

- Gerard Huet has developed a lemmatizer that attempts to tag inflected Sanskrit verbs along with other words. This lemmatizer knows about inflected forms of derived stems which are not apparent in the display of the main stem inflection. It, however, does not attempt to lemmatize verbal forms with preverbs but only invert root forms. The site also provides a long list of the conjugated forms of verb-roots in the present, imperfect, imperative, optative, perfect, aorist and future tenses as a PDF document.
- *Prajna* project of ASR Melkote claims to do module generation and analysis of 400 important Sanskrit roots in three voices (Active, Passive and Impersonal), 10 lakāra, 6 tense and 4 moods,
- Aiba (2004) claims to have developed a Verb Analyzer for classical Sanskrit which can parse Sanskrit verb in Present, Aorist, Perfect, Future, Passive and Causative forms. This site actually works only for some verbs and accepts that the results are not reliable,
- *Desika* project of TDIL, Govt. of India claims to be an NLU system for generation and analysis for plain and accented written Sanskrit texts based on grammar rules of Pāṇini's *Aśṭādhyāyī*. It also claims to have a database based on *Amarakośa* and heuristics based on *Nyāya* & *Mīmāmsā Śāstras* and claims to analyze Vedic texts as well,
- RCILTS project at SC&SS, JNU has reportedly stored all verb forms of Sanskrit in a database,
- *Śābdabodha* project of TDIL, govt. of India claims to be an interactive application to analyze the semantic and syntactic structure of Sanskrit sentences,
- The ASR Melcote website reports that a Sanskrit Authoring System is under development at C-DAC Bangalore. The system is supposed to make making tools for morphological, syntactic and semantic analyses with word split programs for sandhi and *samāsa*.
- Cardona (2004) discussed Pāṇini's derivational system involving aspect of linguistics, grammar and computer science.
- Whitney (2002) listed all the quotable roots of the Sanskrit language together with the tense and the conjugation system.
- Mishra and Jha (2004) describe a module (Sanskrit *Kāraka* Analyzer) for identification and description of *kāraka* according to *Pāṇinian kāraka* formulations.
- Edgren (1885) discussed verb roots of Sanskrit language according to Sanskrit grammarians.
- Joshi (1962) presented linguistic analysis of verb and nouns of Sanskrit language
- Jha and Mishra (2004) proposed *a model for Sanskrit verb inflection identification that would correctly describe verbs in a laukika Sanskrit text.* They presented a module to identify the verb by applying Panini rules in reverse with the help of a relational database.

This module can also be used to identify the types of sentences as active or passive voice with complete reference of the verb.

Present work, which owes a lot to above listed efforts, has some specific features such as:

- The system takes into account the *Pāṅinian* analysis and develops its methodology by applying it into reverse direction.
- It aims at developing a comprehensive strategy so that any *tiṅanta* can be analyzed with the same technique.
- It can be further expanded and modified to recognition and analysis of denominatives
- it is an online servlet-unicode database system with input-output in Unicode only

The front-end of the *tiṅanta* analyzer is as follows -



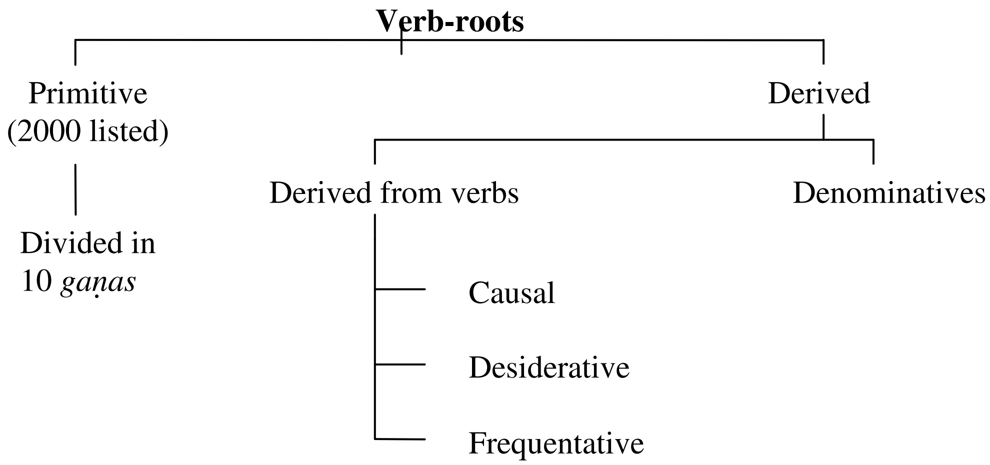
8. Sanskrit Verb-morphology

Verbs have been of central importance to Sanskrit grammarians. *Yāska* insisted so much on them that he propounded that all the nominal words are derived from verb roots. Verbs convey the sense of becoming[10]. Sanskrit follows a well defined process in the formation of *padas*. Both noun *padas (subanta)* as well as verb *padas (tiṅanta)* have to undergo certain inflectional processes in which various nominal or verbal affixes are added to nominal or verbal base word in order to obtain noun and verbal forms. The process is however more than mere addition as there may occur certain morphophonemic changes in the base as well as in the affix in the process resulting in a usable form. The verb forms are derived from verb-roots or *dhātus*.

---

[10] *bhāvapradhānamākhyātam (Yāska, Nirukta)*

These *dhātus* are encoded with the core meaning of the verb. These can be primitive[11] or derived[12]. Primitive verb-roots, which are around 2000 in number, have been listed in a lexicon named *dhātupāṭha*. They are divided in 10 groups called *gaṇas*. All the verb-roots of a group undergo somewhat similar inflectional process. Derived verb-roots may be derived from primitive verb-roots or from nominal forms. Prefixes also play an important role as they can change the meaning of a verb root. These roots then have to undergo various inflectional suffixes that represent different paradigms. In this process, the base or root also gets changed. The chart given on the next page gives an overview of Sanskrit verb roots.



8.1 Derived Verb-roots:

8.1.1 those derived from verb-roots:

- **Causals *(ṇijanta)* -** The causals are formed by adding affix *ṇic* to a primitive verb root. They convey the sense of a person or thing causing another person or thing to perform the action or to undergo the state denoted by the root.

- **Desideratives *(sannanta)* -** Desiderative of a primitive verb root is formed by adding affix *san* to it. It conveys the sense that a person or thing wishes to perform the action or is about to undergo the state indicated by the desiderative form. Any basic verb-root or its causal base may have a desiderative form.

- **Frequentatives *(yaṅanta)* -** Frequentative verbs import repetition or intensity of the action or state expressed by the root from which it is derived. They can be of two types -
    - *Ātmanepada* Frequentative (*yaṅanta*) – affix *yaṅ* is added
    - *Parasmaipada* Frequentative (*yaṅluganta*) – affix *yaṅ* is added but deleted

---

[11] *bhūvādayo dhātavaḥ* (*Pāṇini* 1/3/1)
[12] *sanādyantā dhātavaḥ* (*Pāṇini* 3/1/32)

An illustration is given below of formation of derived verb-roots from a primitive verb root *bhū*.

```
                              --- (+ (ic)   ---- bhāvay (to cause someone or something to be)
bhū ( to be )-----------  --- (+ san)  ---- bubhū( (to desire to be)
                              --- (+ yaṅ)  ---- bobhūya (to be repeatedly)
                              (yaṅ deleted)  ---- bobho/bobhav
```

These derived verb-roots, however, undergo similar operations, with some specifications, to form verb forms.

8.1.2 those derived from nominal words

A large number of Sanskrit verb-forms can be derived from nominal words. These are known as *nāmadhātus* (denominatives). Taking a nominal word as head, various derivational suffixes are added to these to form nominal verb-roots. The sense conveyed by the nominal verb root depends upon the suffix added to it. Yet, denominatives commonly import that a person or thing behaves or looks upon or wishes for or resembles a person or thing denoted by the noun. These denominatives, however, can be innumerable as there is no end to nominal words in Sanskrit.

8.2 Process of formation of Sanskrit verb forms

A Sanskrit verb root may take various forms in different inflectional paradigms. Sanskrit has ten *lakāras*, i.e. four moods (Indicative, Imperative, Optative, and Subjunctive) and six tenses (Present, Imperfect, Perfect, Distant Future, Future and Aorist). The *lakāras* are named in C-V-C format. The first consonant *l* signifies that the suffix has to be replaced by *tin* terminations further. The vowels *a, i, o, u, e, o, r* distinguish one *lakāra* from another. Last consonant, either *ṭ* or *ṅ*, signifies different operations. These *lakāras* are added to the root, as primary suffixes, so that it denotes a meaning in the particular tense or mood indicated by that particular *lakāra*.

Verb inflectional terminations or conjugational suffixes are 18 in number. These are divided in two groups – *Parasmaipada* and *Ātmanepada*, each having 9 affixes – a combination of 3 persons x 3 numbers. Thus each of the 18 terminations expresses the voice, person and number. A verb is conjugated in either *pada,* though some of the roots are conjugated in both. For each different *lakāra,* a root is affixed with these 9 terminations in a single *pada*. Again, there are three voices- Active, Passive and Impersonal. Transitive verbs are used in the Active and Passive voices while intransitive verbs are conjugated in the Active and Impersonal voices. The 18 inflection terminations are basically replacement of the *lakāra* or primary suffix. According to Pāṇini, when a *lakāra* is added to a root, it is replaced by 18 terminations. Thereafter, one of the 18 remains to create a verb form.

For each separate *lakāra*, the 18 *tin* terminations are replaced by other forms, an illustration of the replacement technique of Pāṇini. Thus *ti, tu, tā, t* etc. are the various replacements of same affix tip in the environment of different *lakāras*.

Then we have certain characteristics *(vikaraṇa)* inserted between the root and the termination. This characteristic can vary according to *lakāra* or the class of the verb root. For four of the *lakāras*, we have *śap* as a characteristic – only for four *gaṇas*.

Addition of one or more of 22 prefixes (*upasargas*) to verb roots can result in more variety of forms. Derivative verb roots, both derived from verb roots as well as nominal words, also follow the same process to form verb forms. There can be some specific rules and exceptions in some cases. The following tree gives a rough estimate of all the possible verb-forms of Sanskrit.[13].

VR [2000]
├ **Causal** ((ijanta)
├ Desiderative (sannanta)
└ Frequentative (ya(anta/ya(luganta)
+ One normal form

Denominatives
↓
TAM [10 *lakāras*]
↓

Active Voice	**Passive/Impersonal Voice**
↓	↓
10x9(3x3) forms	**10x9(3x3) forms**
↓	↓
22(+) prefixes	**22(+) prefixes**

The verb roots of different *gaṇas* adopt certain terminations when *tiṅ* affixes are added to them. Consequently, the verb roots of these classes form verbal bases ending in *'a'*. The *tiṅ* affixation also influences the verb root and it undergoes several morpho-phonemic changes, for example, having *guṇa* operation on the end vowel. The verb root can adopt certain more operations resulting in the final verb-form.

bhū + ti(p)
↓
bhū + (ś)a(p) + ti (infixation of characteristic)
↓
bho + a + ti (penultimate vowel gunated)
↓
bhav a ti (*ayādi* sandhi)

As shown in an example, when suffix *tip* is added to verb-root *bhū*, we obtain *bhavati* as the final verb form. This *bhavati* can be analyzed in *bhav + a + ti*. Here *bhav* is the prepared verbal base whereas *a+ti* is the combination of 'characteristic + conjugational affix.' This can be cited as a common analysis of most Sanskrit verb forms. The verbal base of a verb root remains same in all its forms whereas the second combination is common for almost all the roots of a single *gaṇa.* The analysis applies to the first category of derived verb roots as well.

[13] **Mishra Sudhir K., Jha, Girish N.,** 2004, ***Identifying Verb Inflections in Sanskrit morphology*,** In proc. of SIMPLE 04, IIT Kharagpur, pp. 79-81.

9. Analysis of Sanskrit verb forms

9.1 Strategy for regular verb forms

The simplest strategy for regular verb forms can be to store all the possible forms of all the verb roots in any structured form. But given the enormity of Sanskrit verb-roots and the multiplicity of inflectional paradigms, this approach is far from being practical. A better approach may be arrived at by following the analytical method.

As illustrated above, Sanskrit verb forms are a blend of multiple morphemes which contain relevant information. Analytically, we can say that the first element is the conjugational affix that remains at the end of every verb form. These affixes have encoded information of *pada* (though it is determined by the root), *lakāra*, person and number. Thus terminations can serve as the most important thing to let us know about the paradigm information of any verb form. They can be a tool to identify a verb form in a given text. The terminations, as they are basically replacements of 18 original *tiṅ* affixes in different *lakāras*, differ among themselves according to the *lakāra*. However, in each *lakāra* they are similar for all the verb roots of various groups, leaving some exceptions. So, *ti* can be used to identify any verb form of present tense in *parasmaipada*. But some terminations can vary among themselves for a group of *gaṇas*. Then again, the terminations may be changed due to morphophonemic environment, *tā* affix of *luṭ lakāra* changing to *ṭā* with roots like *yaj*.

Further left we have the remaining morphemes of the various characteristics and increments inserted between the verb root and terminations, in the process of their formation explained above. So*, bhvādigaṇa* verb forms, in conjugational *lakāras*, have '*a*'- a result of *śap* characteristic; *svādi* roots have *no*, *nu* or *nv* - all of them remaining morphemes of *śnu*. Some roots like that of *adādi* have no such characteristic sign infixed in them.

Then we have the modified stem of the verb root at the right end of the verb form. The modification can be that of *guṇa*, *vṛddhi* or any other. Generally a root adopts a common stem in all the forms for both the *padas* in conjugational *lakāras*. So, *bhav* is the stem for all the *parasmai* forms in the conjugational *lakāras*. But there are exceptions to it to the extent that four or five types can be found among nine forms of a single *lakāra-pada*.

Here, the first morpheme- the *tiṅ* termination is common among all the verb forms of a particular *pada*-*lakāra*-person-number combination. Second constituent- the characteristic (existing in the form of its remaining morpheme) and increments inserted in between may differ, yet being almost the same in a particular group. The third constituent- the modified verb-root is particular in the strict sense. In the analysis, the recognition of the *tiṅ* termination will identify a word as a verb form and find out its *pada-lakāra*-person and number. The second morpheme can, in many cases, be helpful to recognize the *gaṇa* of a particular root because the characteristics in a *lakāra* are determined by the *gaṇa* that the root belongs to.

Thus the core of the analytical approach is that each *tiṅanta* verb form can be analyzed to form a unique combination of verbal stem + *tiṅ* termination; and we store both of these constituent parts in separate tables. So, when it is to be analyzed, its constituent morphemes are recognized and identified with the help of pre-stored structured data.

An example of this strategy is shown in the table given below. The first column demonstrates the representative verb root of each class. When the verbal affix *tip* is added to each of them, it undergoes certain morphological operations and results in the usable *tiṅanta* verb form listed in

the second column. This is the forward Pāṇinian process. The next column demonstrates the reverse Pāṇinian approach for analysis of verb forms. In the second column every form has *ti* ending. When we remove this ending along with the conjugational affix, we obtain the storable verbal base. Every verb form can be analyzed similarly in ending and remaining verbal base.

Verb-root		**Verb-form**		**Verb-base**
bhū		*bhavati*		*bhav (-ati)*
ad		*atti*		*at(-ti)*
hu		*juhoti*		*juho(-ti)*
div		*dīvyati*		*dīvy(-ati)*
su	+ *ti* →	*sunoti*	- *ti* →	*sun(-oti)*
tud		*tudati*		*tud(-ati)*
chid[14]		*chinatti*		*chinat(-ti)*
tan		*tanoti*		*tan(-oti)*
krī		*krīṇāti*		*krī(-ṇāti)*
cur		*corayati*		*coray(-ati)*

9.2 database tables for verb analysis

The database tables given below demonstrate the structure of storage of all possible verbal bases of a verb root. As a sample data, five verb roots of different *gaṇas* have been taken -

Table of Verbal Bases

root	**gaṇa**	**pada**	**seṭ/ aniṭ/ veṭ**	**lakāra**	**Verbal Bases**				
					Regular	**Causal**	**Desider.**	**Frequentative**	
								Ātmane	**Parasmai**
bhū	*bhvādi*	*parasmai*	*seṭ*	*laṭ/loṭ/vli*	*bhav,*	*bhāvay*	*bubhūṣ*	*bobhūy*	*bobhav ,bobho*
				liṭ	*babhūva*	*bhāvayāñ/m*			
				laṅ	*abhav*	*abhāvay*			
				ali	*bhū*	*bhāv*			
				luṅ	*abhū*	*abībhav*			
ad	*adādi*	*parasmai*	*seṭ*	*laṭ/loṭ/vli*	*ad,at*	*āday*	*jighats*	-	
				liṭ	*jaghāsa, jaghasa jakṣa,āda*	*ādayñ*			
				laṅ	*ād,āt*	*āday*			
				ali	*ad*	*ād*			
				luṅ	*aghas*	*ād*			
hu	*juhotyād*	*para*	*aniṭ*	*laṭ/loṭ/*	*juho,*	*hāvay*	*juhūṣ*	*johūy*	*joho*

[14] *rudh* shows an exceptional behaviour, so *chid* has been taken.

	i	*smai*		*vli*	*juhu,juhv*				
				liṭ	*juhavāñ/m, juhāva, juhuv*				
				laṅ	*ajuho,ajuhu*				
				ali	*hū*				
				luṅ	*ahau*				
div	*divādi*	*para smai*	*seṭ*	*laṭ/loṭ/ vli*	*dīvy*	*devay*	*dideviṣ*	*dedīvya*	-
				liṭ	*didev,didiv*				
				laṅ	*adīvy*				
				ali	*dīv*				
				luṅ	*adev*				

The second table illustrates the structure of the storage of verbal terminations of five *gaṇas* in both *padas* for *laṭ lakāra*. More than one termination in a single box has been separated.

Table of Verbal Affixes

lakāra	**pada/gaṇa**		**I per.**			**II per.**			**III per.**		
			Sing	**Dual**	**Plu.**	**Sing.**	**Dual**	**Plu.**	**Sing**	**Dual**	**Plu.**
laṭ	**parasmai**		*ti/ ati/ oti/ īti*	*taḥ/ ataḥ/ utaḥ*	*nti/ anti /vanti*	*si/ṣi/ asi/ aṣi/o ṣi/osi*	*thaḥ/ athaḥ/ uthaḥ*	*tha/ atha/ utha*	*mi/ āmi/ omi*	*vaḥ/ āvaḥ /uvaḥ*	*āmaḥ / maḥ/ umaḥ*
	ātmane		*te/īte /ūte ute/ īte*	*ete aate/ īyāte/ uvāte*	*ante / ate/ īyate/uvate*	*se/ īṣe/ uṣe ase*	*ethe/ āthe/ īyāthe/ uvāthe*	*adhve / īdhve / udhve*	***e/*** *īye/ uve*	*āvahe/īvahe/uvahe*	*āmahe/īmahe/umahe*
laṅ	**parasmai**		*t/at*	*tām/ atām*	*n/an*	*ḥ/aḥ*	*tam/ atam*	*ta/ ata*	*m/am*	*āva*	*āma*
	ātmane		*ata*	*etām/*	*anta*	*athāḥ*	*ethām*	*adhvam*	*e*	*āvahi*	*āmahi*
luṅ	**parasmai**		*īt*	*iṣṭām*	*iṣuḥ*	*īḥ*	*iṣṭam*	*iṣṭa*	*iṣam*	*iṣva*	*iṣma*

	ātmane		*iṣṭa*	*iṣātām*	*iṣata*	*iṣṭhāḥ*	*iṣāthām*	*idhvam/idhvam*	*iṣi*	*iṣvahi*	*iṣmahi*

For identification and analysis, the suffixes should be given a descending character sequence. So *ti* of *iṣyati* in *bhaviṣyati* cannot create any ambiguity.

10. Problems and possible solutions

- Verb forms which have no mark of termination left in the end are difficult to identify with the proposed module. So *bhava*, *babhūva* and other alike forms are to be stored separately.

- Some forms which are not *tiṅanta* but are similar to them like *bhavati, bhavataḥ* which are singular and dual of *bhū* in present *parasmai* third person, and also locative singular and ablative/relative singular of nominal root *bhavat.* The resolution of ambiguity here will demand involvement of semantic and syntactic analysis.

- Denominatives are formed by deriving verbal base from nominal base with the help of affixes such as *kyac, kāmyac, kyaṣ, yak, kyaṅ* etc. and then adding various verbal terminations to these verbal bases. Thus they undergo same operations and processes as regular and derived verb forms. Still there analysis is difficult due to two reasons. Firstly, nominal bases can be innumerable and thus the above stated strategy of storing the bases of all the nominative verbal bases is impossible in this case. One has to follow the rule based analytical approach. The verbal terminations can be determined with the help of affix tables as denominatives are affixed with same verbal affixes. The remaining base, however, has to be analyzed in order to infer the nominal base of that denominative. As there are some common rules to derive the verbal stem from nominal base, we can develop an analysis rule based module to identify the nominal root and can find its meaning with the help of a lexicon.

- Addition of prefixes to the verbal bases may cause morphological as well as semantic change to a verbal form. To identify one or more prefix in a verbal form, all the prefixes have to be stored in a database table along with their meaning. The system will have to check the input verbal form from left to identify single or combined prefixes. A prefix can happen to completely modify the meaning of a verb. So, creating a separate table that stores the altered meanings of various roots, when affixed with certain prefix, may be helpful in this case.

11. Conclusion

The proposed strategy to analyze Sanskrit verb forms in given text is different from existing works in many ways. It works with a reverse Pāṇinian approach to analyze *tiṅanta* verb forms into there verbal base and verbal affixes. The methodology accepted to create database tables to store various morphological components of Sanskrit verb forms is clearly in line with the well defined and structured process of Sanskrit morphology described by Pāṇini in his *Aṣṭādhyāyī*. It comprehensively includes the analysis of derived verb roots also. Even in the case of verb roots derived from nominal words, the table of affixes can provide assistance in order to separate the denominative verbal base from the verbal terminations.

References

1. BHARATI, AKSHAR, VINEET CHAITANYA & RAJEEV SANGAL, 1991, *A Computational Grammar for Indian languages processing*, Indian Linguistics Journal, pp.52, 91-103.
2. BHARATI, AKSHAR, VINEET CHAITANYA AND RAJEEV SANGAL, 1995, *Natural Language Processing: A Paninian Perspective*, Prentice-Hall of India, New Delhi.
3. CARDONA, GEORGE, 1967, *Pāṇini's syntactic categories*, Journal of the Oriental Institute, Baroda (JOIB) 16: 201-15
4. CARDONA, GEORGE, 1988, *Panini: his work and its tradition (vol.1)*, Motilal Banarasidas, Delhi,
5. CARDONA, GEORGE, 2004, *Some Questions on Pāṇini's Derivational system*, procs of SPLASH, iSTRANS, Tata McGraw-Hill, New Delhi, pp. 3
6. JURAFSKY DANIEL AND JAMES H. MARTIN, 2000, *Speech and Languages Processing,* Prentice-Hall, New Delhi
7. EDGREN , A. H., 1885, *On the verbal roots of the Sanskrit language and of the Sanskrit grammarians*, Journal of the Americal oriental Society 11: 1-55.
8. HUET, G'ERARD, 2003, *Towards Computational Processing of Sanskrit, Recent Advances in Natural Language Processing*, Proceedings of the *International Conference ICON*, Mysore, India
9. JHA, GIRISH N. et al., 2006, *Towards a Computational analysis system for Sanskrit*, Proc. of *first National symposium on Modeling and Shallow parsing of Indian Languages* at Indian Institute of Technology Bombay, pp 25-34
10. JHA, GIRISH N, 2003 *A Prolog Analyzer/Generator for Sanskrit Noun phrase Padas*, Language in India, volume-3,
11. JHA, GIRISH N, 2004, *Generating nominal inflectional morphology in Sanskrit*, SIMPLE 04, IIT-Kharagpur Lecture Compendium, Shyama Printing Works, Kharagpur, WB. Page no. 20-23.
12. JHA, GIRISH N., 1993, *Morphology of Sanskrit Case Affixes: A computational analysis*, M.Phil dissertation submitted to J.N.U., New Delhi
13. JHA, GIRISH N., 2004, *The system of Panini*, Language in India, volume4:2,
14. JOSHI, S. D., 1962, *Verbs and nouns in Sanskrit*, Indian linguistics 32 : 60- 63.
15. KAPOOR, KAPIL, 1985, *Semantic Structures and the Verb: a propositional analysis*, Intellectual Publications, New Delhi
16. MISHRA SUDHIR K., JHA, GIRISH N., 2004, *Identifying Verb Inflections in Sanskrit morphology*, In proc. of SIMPLE 04, IIT Kharagpur, pp. 79-81.

17. MISHRA, SUDHIR K & JHA, GIRISH N, 2004, *Sanskrit Karaka Analyzer for Machine Translation*, In SPLASH proc. of iSTRANS, Tata McGraw-Hill, New Delhi, pp. 224-225.
18. MITKOV RUSLAN, *The Oxford Handbook of Computational Linguistics*, Oxford University Press.
19. NARAYAN MISHRA, 1996, (ed). *Kashika of Pt.Vamana and Jayaditya* ,Chaukhamba Sanskrit sansthan, Varanasi
20. NOOTEN, B. A. VAN, *Pāṇini's replacement technique and the active finite verb*, University of California, Berkeley.
21. SHARMA, RAMA NATH, 2003, *The Aṣṭādhyāyī of Pāṇini*, Munshiram Manoharlal Publishers Pvt. Ltd., Delhi.
22. SHASTRI, BHEEMSEN, *Laghusiddhantakaumudi (1st part)*, Bhaimee Prakashan, 537, Lajapatrai Market, New Delhi
23. SHASTRI, SWAMI DWARIKADAS, 2000, 'The Mādhavīya Dhātuvṛtti by Sāyaṇacārya', Tara Book Agency, Varanasi.
24. SUBASH & JHA, GIRISH N., 2005, *Morphological analysis of nominal inflections in Sanskrit*, presented at Platinum Jubilee International Conference, L.S.I. at Hyderabad University, Hyderabad, pp-34.
25. SUBASH, 2006, *Machine recognition and morphological analysis of Subanta-padas*, M.Phil dissertation submitted to J.N.U., New Delhi.
26. UPADHYE, P.V., 1927, *Dhāturūpacandrikā*, Gopal Narayen & Co, Bombay.
27. WHITNEY, W.D., 2002, *History of Sanskrit Grammar*, Sanjay Prakashan, Delhi.

Web References:

- IIIT, Hyderabad, http://www.iiit.net/ltrc/Publications/Techreports/tr010/anu00kbcs.txt (accessed: 22nd April 2007).
- Peter M. Scharf and Malcolm D. Hyman, http://sanskritlibrary.org/morph/ (accessed: 12 August 2006).
- Huet's site http://sanskrit.inria.fr/
- Prajna system, ASR Melcote, http://www.sanskritacademy.org/Achievements.htm
- Aiba, Verb Analyzer for classical Sanskrit, http://wwwasia.human.is.tohoku.ac.jp/demo/vasia/html/
- Desika, TDIL, Govt. of India, http://tdil.mit.gov.in/download/Desika.htm
- RCILTS, JNU, http://rcilts.jnu.ac.in
- Shabdabodha, ASR, Melcote, http://vedavid.org/ASR/#anchor2

PHONOLOGICAL OVERGENERATION IN PANINIAN SYSTEM

Malhar Kulkarni
IIT, Powai, Mumbai- 400076
malhar@iitb.ac.in

M.M.Vasudevashastri
Abhyankarashastri Pathashala,
Pune.

ABSTRACT

In this paper an attempt is made to study the problem of overgeneration that is caused by the application of the system of *Pāṇini* The system of *Pāṇini* is made up of certain rules stated by him and his commentators namely, *Kātyāyana* and *Patañjali*. These rules are supposed to produce the forms that are used in the language, i.e. Sanskrit. However, sometimes the technical application of these rules produces such forms which are not actually used in the language. In fact, sometimes it is beyond human capacities to use such forms. In the present paper two such cases dealing with the phonological overgeneration are studied and possible solutions are proposed to avoid the problem.

1. INTRODUCTION:

It has been demonstrated by Kiparsky and Staal(1988) how Paninian system functions on four levels, namely, semantic, deep structure, surface and phonological. This system however sometimes over-generates in perhaps, some of these levels. Of course *Pāṇini* (P) has no doubt laid down certain constraints with the help of which the system produces supposedly un-over-generated forms. Prince and Smolensky (2002), have devoted a section on Panini's theorem of constraint ranking (5.3) Of course our judgement regarding the over-generativeness of a rule in the *Aṣṭādhyāyī*(A), it must be admitted here, is based entirely upon whatever evidance in the form of pre-paninian literature available to us.

2. PHONOLOGICAL OVER-GENERATION

This paper is devoted to phonological over-generation that still happens with all the possible constraints applying. There are two aspects that are studied in this paper,

(1) Nasalization and (2) Phonetic doubling

2.1. Nasalization:

8.4.45 states that *yar*[1] occurring at the end of a *pada*, is optionally, (*preferably*, according to Kiparsky1980:1) substituted by the nasal, if a nasal follows.

(1)
etad murāriḥ
= *etan murāriḥ* / *etad murāriḥ* . . . 8.4.45

Kātyāyana(K) has added a *Vārttika*(V) on this rule, to the effect that this nasalization takes place permanently if the following nasal is a part of a suffix

(2)
tad + maya
= *tan-maya* . . . 8.4.45 + K's V.
= *tanmaya*

2.1.1. *Enviornment for nasalization:*

However, if we look at the way P has stated this rule, we have to take into account following table which shows clearly all possible environments in which this rule should apply and the possible results in the form of substitution of a nasal consonant. The top row and the left column, in the table, show the possible environment. The bottom row shows the resultant nasal consonant in place of the consonant written in the same column in top row. Thus for

[1] These phonemes are- all the stops including the nasals, semi vowels(y, r,l,v) and sibilants except h.

instance,

[..y]	+ [n..] / [m..] /	...	8.4.45 + K's V
= [..y#]	+ [n..] / [m..] /		

Table 1 shows that any consonant mentioned in the top row occuring at the end of a *pada* and followed by any of the nasal consonants mentioned in the left hand column, is substituted by the nasal consonant shown in the bottom row. # mark is used to show the nasal feature in the bottom row. * shows that these substitutions are not attested in Sanskrit. The order of sounds followed by P in his *pratyāhāra sūtra*s is maintained here.

There are certain sounds in this table which are directly not applicable for this operation as they never occur at the end of a *pada* in Sanskrit. Such sounds are- y,l,, *ì , ñ , jh, bh, gh, ḍh,dh, kh, ph, ch*. Some grammatical entries do end in some of these sounds and hence it can be argued that by applying operations related to 0 suffix, one can generate *pada*s with these sounds at the end. However, this argument does not hold valid as in the case of these consonants, the other rules namely, 8.2.30, 8.2.39 etc. will substitute them with the other consonants.

Thus consider the following example-

(3)

gumph		...	*Dhātupāṭha* 6.31
gumph	+ *kvip*	...	3.2.178
gu ph	+ *kvip*	...	6.4.24
gu ph	+ 0	...	6.1.67
= *guph*			
guph	+ *su*	...	4.1.1,2
guph	+ 0	...	6.1.68
gub		...	8.2.39
gub/*gup*		...	8.4.56
= *gub* / *gup*			

In the same way, other consonants will be substituted.

2.1.2. *Overgenerated nasalization:*

Now the rule, applied to all the remaining consonants should also apply to the following example-

(4)

catur mukha	......	8.4.45
catu ṇ mukha		
= *catu ṇmukha*		

However, this resultant form is not acceptable in Sanskrit. This is clearly an over- eneration.

8.4.58 states the substitution of a nasal in place of an *anusvāra* when followed by almost same consonants(called as *yay* by P) mentioned in the top row of Table 1 above except the last three. The rule can be shown as-

[... *anusvāra*]	+ [*yay*...]
= [... nasal]	+ [*yay*...]

Thus by applying this rule we get forms like *kaṇtha, aìkita, guṃphita* etc. Consider however, the following example-

(5)

kuṇḍam rathena		
kuṇḍaṃ rathena	...	8.3.23
kuṇḍaṇ rathena	...	8.4.58

The resultant form here is not acceptable in Sanskrit. This is again over-generation.

One may argue about redundancy being the feature of use of the *pratyāhāra*s in the metalangauge of However, the tradition has taken pains in creating a constraint to check such forms in the form of statements in this regard. Pa in the context of the above example says-

rephoṣmaṇām savarṇā na santi [2]. (the sounds r and the sibilants do not have any homogenious(nasal))

There are at least some constraints in the form of statements of the later commentators to check the overgeneration as shown above. However, in the case of phonetic doubling mentioned below, we see hardly any constraint to check the overgeneration.

2.2. Phonetic doubling:

P in his A has dealt with the process of reduplication at three places; (i) 6.1.1-12[3], (ii) 8.1.1-15, (iii) 8.4.46-52. (i) deals with the reduplication of verbal roots in the forms of present as well as perfect tense and also in forming complex verbal roots such as desiderative and frequentative. In a nutshell, this reduplication applies to the *aìga* in Paninian terminology. (ii) deals with the reduplication of the entire *pada*. The last section in the A mentioned above, deals with the reduplication of the consonants. The paninian tradition has augmented the existing set of rules laid down by P in this section, in the form of *Vārttika*s (maily written by K) in this regard and the later tradition has interpreted certain statements of *Patañjali*(Pa) in such a manner that the resultant forms can only be termed as over-generated ones. The later paninian tradition has done this exercise at many places and has come up with such overgenerated

[2] *VyākaraëaMahābhāṣya* of *Patañjali*, 2001, Vol.1, p 130.

[3] More recently, Kiparsky in a forthcoming article available on his webpage, has discussed it.

Table 1: Consonants and their substitutes according to 8.4.45

	y	v	r	l	ñ	m	ṅ	ṇ	n	jh	bh	gh	ḍh	dh	j	b	g	ḍ	d	kh	ph	ch	ṭh	th	c	ṭ	t	k	p	ś	ṣ	s
ṅ																																
ñ																																
ṇ																																
n																																
m																																
	y	v	ṇ	l	ñ	m	ṅ	ṇ	n	ñ	m	ṅ	ṇ	n	ñ	m	ṅ	ṇ	n	ṅ	m	ñ	ṇ	n	ñ	ṇ	n	ṅ	m	ñ	ṇ	n
	#	#	*	#																										*	*	*

Table 2: Enviornment for Phonetic doubling

1	**2**	**3 Consonant Reduplicated**	**4**	Rule of Panini
vowel	*r/h*	*yar*	—	8.4.46
—	vowel	*yar*	No vowel	8.4.47
vowel	*Yaṇ*	*may*	—	K & Pat on 8.4.47
vowel	*may*	*yaṇ*	—	As above
—	*Ś ar*	*khay*	—	As above
—	*khay*	*Ś ar*	—	As above

forms. Tne such extreme cases are presented in this paper and an attempt is made to study the approach of the Paninian system to handle this phenomenon.

(6) *putrādinī tvam asi pāpe*

(Oh! son-eater woman, shame on you!)

puttrādinī sarpiṇī

(she-snake is son-eater.)

In this case, t is seen reduplicated alongwith the change in the meaning. This case is noted by 8.4.48.

2.2.1. Enviornment for Phonetic doubling:

In the same section, some other phonemes are also noted for their reduplicated occurrence. K and pat have also noted down this tendency in some other phonemes. These phonemes are- same mentioned in fn 2. In table 2 they are referred to as *yar*, as used by P. In the table 2, these rules are explained with all details, namely environments- prior and posterior

Here 1 , 2 , 4 refer to the environment for phonetic doubling. The order indicates the positions of these environments and the position of the phoneme reduplicated. The examples for these two rows are-

(7) *h**aryy**anubhavaḥ*
(h *a-r-y anubhavaḥ* > phonetic doubling of y)

(8) (a)*rāmā**tt***
(*rām ā-t-*(no vowel) > phonetic doubling of t)

(b) *s**udhdh**yupāsyaḥ*
(*s-u-dh-y upasyah* > phonetic doubling of y).

2.2.2. K and Pa on the environment for phonetic doubling:

While commenting upon 8.4.47, K notes- *dvirvacane yaṇo mayaḥ* . On this Pa has a two fold comment. He says-

dvirvacane yaṇo maya iti vaktavyam.
Kim udāharaṇam yadi yaṇa iti
pañcamī maya iti ṣaṣṭhī
ulkkā valmmikam ity udahāraṇam. Atha maya
iti pancamī yaṇa iti ṣaṣṭhī
dadhyyatra madhvvatrety udāharaṇam

This means- In the rules dealing with the process of phonetic doubling, the words yaṇo mayaḥ should be stated. What is the example ? If *yaṇaḥ* (*yan* is *y, v,r, l*) is taken to be ablative and *mayaḥ* (*may* is all stops except nasal palatal) is taken to be genitive, then the examples are –

(9) *u**lkk**ā / va**lmm**ikam*

and if *mayaḥ* is taken to be ablative and *yaṇaḥ* is taken to be genitive, then the examples are-

(10)*da**dhyy**atra / ma**dhvv**atra.*

Same argument is applied to another statement of K, namely *ś araḥ khayaḥ*[4] which provides us with the following examples-

(11) ***sthth**ālī / **sthth**ātā*

(12) *va**tss**aḥ / **kss**īram / a**pss**arāḥ*

This way of interpreting the statements of K on the rules of P becomes a peculiar feature of the system of paninian grammar. Later tradition of paninian grammar thus by interpreting statements of K and Pa and

[4]This statement means that *khay* is reduplicated if it occurs after *ś*ar and *ś*ar is reduplicated if it occurs after *khay*. *śar* stands for all the sibilants except h and *khay* stands for all the voiceless stops.

P have noted down forms which we here address as overgenerated forms.

We note that this feature is also noted by non-paninian systems such as *Kātantra*. A commentary on *PrakriyāKaumudī* namely *Prakāś a* notes that according to *Kātantra* school the phonetic doubling in a particular case will give rise to only 32 forms and not more[5].

2.2.3. Twice Occurences of same consonant in Sanskrit

It is noteworthy to study the structure of the consonant cluster in Sanskrit. A list of such clusters is available in Coulson Michael, 2003, p 22-24. We concentrate on a cluster of two consonant of same phonetic value. In other words, we concentrate on the twice occurrence of the same consonant. In the table 3, a list of such consonant clusters is provided. Table 3 shows us the consonants which can have twice occurrence without applying the rules of phonetic doubling.

Table 3: Twice occurences of same consonant

k	(i)Final + initial of the next word (ii)Prefinal
g	Final + initial of the next word
c	Final + initial of the next word
j	Final + initial of the next word
t	Final + initial of the next word
d	Final + initial of the next word
p	Final + initial of the next word
b	Final + initial of the next word
ṅ	Final + initial of the next word
n	(i) Final + initial of the next word (ii)Pre-final
ṇ	Final + initial of the next word
m	Final + initial of the next word
ś	Final + initial of the next word
ṣ	Final + initial of the next word
s	Final + initial of the next word

A careful glance at table 3 will point out that all these consonants fall in the domain of the application of the phonetic doubling rules mentioned above in Table 2. Therefore, if the rule for phonetic doubling is applied to these already existing two consonants, we get three same consonants occurring one after another. Such a form is noted to exist optionally by P in the case of consonants except nasals by the 8.4.65.

[5] *PrakriyāKaumudī*, 2000, Vol.I, p 158.

2.2.4. Generation of Phonetic doubling in later tradition:

A 17^{th} century grammar text, *Vaiyākaraṇa-SiddhāntaKaumudī* (VSK) records following cases of phonetic doubling-

(13) *rāmātt rāmādd.*/ VSK 206, *dvitve rūpacatuṣṭayam.* (in the forms *rāmāt* and *rāmād*, after applying the rules of phonetic doubling we get 4 forms).

(14) *aidhidhvam* / VSK 2258, *dhaḍhayor vasya masya ca dvitvavikalpātṣoḍaṣarūpāṇi.* (by reduplicating v and m when immediately before *dha* and *ḍha* we get 16 forms.)

(15)*saṃskartā* / VSK 138, *anusvāravatām anusvārasyāpi dvitve dvādasa.* (after reduplicating the *anusvāra* in the forms already containing it, we get 12 forms).

(16) *gavāk* / VSK 443

Cases (15) and (16) deserve a special attention as they pose a problem.

2.2.4.1. 2.2.4.1 Generation of Phonetic doubling in the forms of saṃskartā

(15) *saṃskartā* - This word is formed in the following way[6]-

sam + kartā		
sam + s- kartā	...	6.1.134
sar + s-kartā	...	8.3.5
saṃr + s-kartā	...	8.3.2 / 8.3.4
saṃs + s-kartā	...	8.3.15

Along with this form there is an optional form that is available in which in place of *ṃ* there occurs an *anusvāra*. In the following two tables (Table 4 and Table 5), forms with *ṃ* and *anusvāra* are presented.

In Table 4 and 5, we see phonetic doublings of s , t , k and more problematically of the *anusvāra*. This phonetic doubling of *anusvāra* is based on the argument of K that *ayogavāha*[7] s are to be included in the *pratyāhāra aṭ* as well as *śar* by the statement-*ayogavāhānam aṭṣuṇatvam śarṣu jaś tvaṣatve.*

[6] I have to turn to Devnagari fonts for these two case to stress the amount of problem.

[7] The term *ayogavāha* refers to *anusvāra, visarga, jihvāmulīya and upadhmānīya*, Vy\=akaraëaMahābhāṣya of Patañjali, 2001, Vol.1, p 132.

Table 4: Forms of *saṃskartā* with a first nasal vowel

Forms	Explanation
संस्कर्ता	Deletion of first *s* - comment of Pa-*samo*
संस्स्कर्ता	*vā lopam ity eke Mbh.* on P.8.3.5
संस्स्स्कर्ता	Phonetic doubling of 1st *s* by P.8.4.47
संस्क्कर्ता संस्स्क्कर्ता संस्स्स्क्कर्ता	Phonetic doubling of *k* – *śaraḥ khayaḥ* referred to in Table 2 above.
Forms: संस्कर्त्ता संस्स्कर्त्ता संस्स्स्कर्त्ता संस्क्कर्त्ता संस्स्क्कर्त्ता संस्स्स्क्कर्त्ता	Phonetic doubling of *t* in all the above 6 forms by *dvir vacane yaṇo mayaḥ*
Forms: संस्क्कर्त्त्ता संस्स्क्कर्त्त्ता संस्स्स्क्कर्त्त्ता	2nd Phonetic doubling of *t* in the first 6 forms by *dvir vacane yaṇo mayaḥ*
Forms: संस्कर्ताँ संस्स्कर्ताँ संस्स्स्कर्ताँ संस्क्कर्ताँ संस्स्क्कर्ताँ संस्स्स्क्कर्ताँ संस्कर्त्ताँ संस्स्कर्त्ताँ संस्स्स्कर्त्ताँ संस्क्कर्त्ताँ संस्स्क्कर्त्ताँ संस्स्स्क्कर्त्ताँ संस्कर्त्त्ताँ संस्स्कर्त्त्ताँ संस्स्स्कर्त्त्ताँ संस्क्कर्त्त्ताँ संस्स्क्कर्त्त्ताँ संस्स्स्क्कर्त्त्ताँ	Nasalization of the final vowel by 8.4.57 in all the 12 forms mentioned above. In all, we have 24 forms in this row.

2.2.4.2. 2.2.4.2 Generation of Phonetic doubling in the forms of Gavāk

The *Paninian Dhātupāṭha* notes that the root *añc* is used in two senses viz. *gati*(to go) and *pūjana*(to worship/ respect). In the sense of 'one who goes to a cow' and in the sense of 'one who worships a cow', the derivations that take place according to the rules of A are shown in the table 6 and table 7 respectively.

In the above tables, the underlined forms are the forms of a noun derived from a verbal root. Note that the difference in the forms in these tables is a mere n which

Table 6: Derivation of *Gavāk* (one who goes to a cow)

go	+ *añc*	...	(in the sense of to go)
go	+ *añc* + *kvin*	...	A. 3.2.59
go	+ *ac*	...	A 6.4.24, A.6.1.67
goc	/ *go ac*	...	A 6.1.123
goc	/ *gavāc* / *go ac*	...	A.6.1.109,122,123,

Table 7: Derivation of *Gavāñc* (one who worships a cow)

go	+ *añc*	...	(in the sense of to go)
go	+ *añc* + *kvin*	...	A. 3.2.59
go	+ *añc*	...	A 6.1.67
goñc	/ *gava añc*	...	A 6.1.123
goñc	/ *gavāñc* / *go añc*	...	A.6.1.109,122,123,

has brought about a sea of change in the meaning as well as the form itself. That is why P has noted them with all their variations.

When we take these 6 forms as the base and start adding the *sup* terminations, we get tables 8 and 9 for these two tables. Tables 8 and 9 correspond to Tables 6 and 7 mentioned above. These are the forms in neuter gender. There are certain specific processes for neuter forms. That is why they are selected here. In these tables, in each slot, there are many optional forms shown. They result out of the optional application of the rules namely, 6.1.109, 6.1.122 and 6.1.123.

The final square in Table 9 has got 9 forms. The last 3 forms are a result of the application of the statement of K[8] according to which the 1^{st} class consonant is replaced by the 2^{nd} class consonant of the same class. Thus we see here k is replaced by *kh*. These are the forms, we can say on the authority of A and K, which are actually spoken by people. So far there is no problem. When we apply the rules of phonetic doubling of certain consonants to these abovementioned Table 8 and 9, we start facing a problem.

2.2.4.3. Effects of phonetic doubling on forms in Tables 8 and 9:

In Table 10 and 11 (as shown in the appendix), the reduplicated forms of the forms mentioned in Table 8 and 9 respectively are presented.

In the table 10 (as shown in Appendix), we note that the phonetic doubling of *k, g, ñ, , y, m* has increased the number of forms (which are indicated in each square).

[8] *cayo dvitīyā śari pauṣkarasādeḥ* / on 8.4.48

The reasoning for the phonetic doubling of *k, g, ñ* is 8.4.47. The reasoning for the phonetic doubling of y and m is the same as mentioned in Table 5 namely, *dvirvacane yaṇo mayaḥ*. We also note that there is a phonetic doubling of even a *visarga* in certain forms. The reasoning for this phonetic doubling is same as mentioned after Table 5, namely, inclusion of *visarga* in the *pratyāhāra yar*. Also there is nasalization which is marked by a sign on certain forms which has added those many forms.

We note that in the table 11 (as shown in the Appendix) the following consonants apart from the ones mentioned in Table 7 are reduplicated- *ì*, *ṣ*. The reasoning for phonetic doubling of *ì* is 8.4.47 and for *ṣ* is the one mentioned in Table 4. namely, *ś araḥ khayaḥ*. We also note that in some forms even the *visarga* is reduplicated like in the previous table. In Table 10 and 11 (as shown in the Appendix), we also note that in some forms three consonants are simultaneously reduplicated. We also see that nasalization is marked with the sign in some forms.

Thus if we compare the tables 10 and 11 (as shown in the Appendix) statistically we come up with the following picture-

Unduplicated	Duplicated + Nasalized
49 (Table 8)	196 (Appendix: Table 10)
69 (Table 9)	267 (Appendix: Table 11)

If we are adopting the Paninian framework for generating forms by machine we will face similar problems if we apply the rules of phonetic doubling .

3. PROPOSED SOLUTION :

This overgeneration of forms is caused by-

(i) redundency of the *pratyāhāra*.

(ii) application of the rules of phonetic doubling mechanically.

(iii) application of statements and interpretations of later paninian commentators.

To solve this problem we propose the following:

If we are going to apply the rules of phonetic doubling we must make a rule that -

R1 The visarga should never be reduplicated.

R2 An anusvara should never be reduplicated.

R3 The rule of phonetic doubling should not be applied more than once to one consonant.

R4 The rule of phonetic doubling should not be applied to more than one consonant simultaniously.

In order to remove the redundency, we have to rely upon the statements of the later comentators and take note of their statements and modify the rule accordingly.

4. ACKNOWLEDGEMENT

I wish to express gratitude to my students Ms.A.Ajotikar, Ms.T.Aajotikar and Ms.Sarnaik, at the Abhyankarashastri Pathashala, Pune for helping me type out the forms in tables presented in this paper. I also wish to express my gratitude to all the scholars who made suggestions and remarks which helped improve this article immensely. I wish to thank Prof. Kiparsky for providing me with the details of the reference of one of his forthcoming publications. I wish to thank also my student Ms. Chatali Dangarikar for helping me format the text of this article.

5. REFERENCES

K. V. Abhyankar, editor. 1943. *Vyākaraṇa-Mahābhāṣya of Patañjali with Marathi Translation (7volumes)*. D.E.Society. reprint, Sanskrit Vidya Parisamstha, Pune, 2007.

Prof. Balshastri, editor. 2001. *Vyākaraṇa-Mahābhāṣya of Patañjali, alongwith Pradīpa, Udyota and Śabdakaustubha,*. Pratibha Prakashan, New Delhi. edited originally by Shri. GuruprasadShastri, reedited by Prof. Balshastri.

B.K.Dalai, editor. 2007. *Phonology, in Studies in Sanskrit Linguistics,*. Bharatiya kala Prakashan, New Delhi.

Otto Böhtlingk, editor. 1998. *Pāṇinis Grammatik*. Motilal Benarsidass, New DElhi, 1^{st} indian edition edition. Aṣṭādhyāyī edited and translated into German.

George Cardona and Dhanesh Jain, editors. 2003. *Indo-Aryan Languages*. Number 11 in Routledge Language Family Series. New Fetter Lane, London EC4P4EE.

George Cardona. 2004. *Recent researches in Paninian studies*. Motilal Benarsidas, , Delhi, 2nd revised edition edition.

Michael Coulson. 2003. *Teach yourself Sanskrit*. Hodder Headline Ltd., revised by prof.richard gombrich and dr james benson edition. 1st published in 1976.

Dr. GirijaKumar Dixit, editor. 1987. *Paribhāśenduśekhara of Nāgeśa, alongwith the commentary Sarvamaṅgalā,*. Sampurnananda Sanskrit University, varanasi edition.

Dr. Sitaramashastri, editor. 1996. *Bṛhacchabdenduśekhara of Nāgeśa (in 3 parts)*. Sampurnananda Sanskrit University, Varanasi, 1^{st} edition edition.

G.Caturveda and P.Bhaskar, editors. 1987. *Vaiyākaraṇa-Siddhānta-Kaumudī, alongwith Bāla-Manoramā and Tattvabodhinī,*. Motilal Benarsidass.

Paul Kiparsky and Staal J.F, 1988. *Modern Studies in Sanskrit*, chapter Syntactic and Semantic relations in Panini.

Paul Kiparsky. 1980. *Panini as a variationist*. Centre of Advanced Study in Sanskrit,University of Poona, in collaboration with MIT press,, Cambridge(Massachusetts, U.S.A.) and London (England).

Paul Kiparsky. (forthcoming). Reduplication in stratal ot. Available at: http://www.stanford.edu/ kiparsky/Papers/reduplication.pdf.

Pt. Muralidhar Mishra, editor. 2000. *Prakriyākaumudī with Prakāśa (in 3 parts)*. Sampurnanada Sanskrit University, Varanasi.

Abbreviations

A	-	*Aṣṭādhyāyī*
K	-	*Kātyāyana*
P	-	*Pāṇini*
Pa	-	*Patañjali*
V	-	*Vārttika*
VSK	-	*Vaiyākaraṇa-Siddhānta-Kaumudī*

Appendix: Table 10:Phonetic doubling in the declension of *Gavāc*

	1	2	3
1	गोऽक्क् गोऽग्ग् गोअक्क् गोअग्ग् गवाक्क् गवाग्ग् 6	गोची	गोऽञ्चि गोऽञ्ञ्चि गोऽञ्चि गोअञ्चि गोअञ्ञ्चि गोअञ्चि गवाञ्चि गवाञ्ञ्चि गवाञ्चि 9
2	गोऽक्क् गोऽग्ग् गोअक्क् गोअग्ग् गवाक्क् गवाग्ग् 6	गोची	गोऽञ्चि गोऽञ्ञ्चि गोऽञ्चि गोअञ्चि गोअञ्ञ्चि गोअञ्चि गवाञ्चि गवाञ्ञ्चि गवाञ्चि 9
3	गोचौ	गोऽग्भ्याम् गोऽग्ग्भ्याम् गोऽग्भ्भ्याम् गोअग्भ्याम् गोअग्ग्भ्याम् गोअग्भ्भ्याम् गवाग्भ्याम् गवाग्ग्भ्याम् गवाग्भ्भ्याम् गोऽग्भ्भ्याम् गोऽग्भ्याम्म् गोऽग्भ्भ्याम्म् गोअग्भ्भ्याम् गोअग्भ्याम्म् गोअग्भ्भ्याम्म् गवाग्भ्भ्याम् गवाग्भ्याम्म् गवाग्भ्भ्याम्म् गोऽग्ग्भ्याम्म् गोऽग्ग्भ्भ्याम्म् गोअग्ग्भ्याम्म् गोअग्ग्भ्भ्याम्म् गवाग्ग्भ्याम्म् गवाग्ग्भ्भ्याम्म् 24	गोऽग्भिः गोऽग्ग्भिः गोऽग्भिःः गोऽग्ग्भिःः गोअग्भिः गोअग्ग्भिः गोअग्भिःः गोअग्ग्भिःः गवाग्भिः गवाग्ग्भिः गवाग्भिःः गवाग्ग्भिःः 12
4	गोचे	Same as above 24	गोऽग्भ्यः गोऽग्ग्भ्यः गोऽग्भ्भ्यः गोऽग्ग्भ्भ्यः गोअग्भ्यः गोअग्ग्भ्यः गोअग्भ्भ्यः गोअग्ग्भ्भ्यः गवाग्भ्यः गवाग्ग्भ्यः गवाग्भ्भ्यः गवाग्ग्भ्भ्यः गोऽग्भ्यःः गोऽग्ग्भ्यःः गोऽग्भ्भ्यःः गोऽग्ग्भ्भ्यःः गोअग्भ्यःः गोअग्ग्भ्यःः गोअग्भ्भ्यःः गोअग्ग्भ्भ्यःः गवाग्भ्यःः गवाग्ग्भ्यःः गवाग्भ्भ्यःः गवाग्ग्भ्भ्यःः 24
5	गोचःः	Same as above (24)	Same as above 24
6	गोचःः	गोचोःः	गोचाम्म्
7	गोचि	गोचोःः	गोऽक्षु गोऽक्क्षु गोऽक्ष्षु गोऽक्क्ष्षु गोऽक्षुं गोऽक्क्षुं गोऽक्ष्षुं गोअक्षु गोअक्क्षु गोअक्ष्षु गोअक्क्ष्षु गोअक्षुं गोअक्क्षुं गोअक्ष्षुं गवाक्षु गवाक्क्षु गवाक्ष्षु गवाक्क्ष्षु गवाक्षुं गवाक्क्षुं गवाक्ष्षुं गोऽक्क्ष्षुं गोअक्क्ष्षुं गवाक्क्ष्षुं 24

Appendix: Table 11:Phonetic doubling in the declension of *Gavāñc*

गोऽञ्च् गोअञ्च् गवाञ्च् पूजार्थे द्वित्व			
1	गोऽङ्ङ् गोअङ्ङ् गवाङ्ङ्	गोऽञ्चौ गोअञ्चौ गवाञ्चौ	गोऽञ्चः गोऽञ्च्चः गोऽञ्चः गोअञ्चः गोअञ्च्चः गोअञ्चः गवाञ्चः गवाञ्च्चः गवाञ्चः 9
2	गोऽङ्ङ् गोअङ्ङ् गवाङ्ङ्	गोऽञ्चौ गोअञ्चौ गवाञ्चौ	गोऽञ्चः गोऽञ्च्चः गोऽञ्चः गोअञ्चः गोअञ्च्चः गोअञ्चः गवाञ्चः गवाञ्च्चः गवाञ्चः 9
3	गोऽच्चा गोऽञ्चा गोऽञ्च्चा गोअच्चा गोअञ्चा गोअञ्च्चा गवाच्चा गवाञ्चा गवाञ्च्चा 9	गोऽङ्भ्याम् गोऽङ्ङ्भ्याम् गोऽङ्भ्य्याम् गोअङ्भ्याम् गोअङ्ङ्भ्याम् गोअङ्भ्य्याम् गवाङ्भ्याम् गवाङ्ङ्भ्याम् गवाङ्भ्य्याम्	गोऽङ्भिः गोऽङ्ङ्भिः गोऽङ्ङ्भिः गोअङ्भिः गोअङ्ङ्भिः गोअङ्ङ्भिः गवाङ्भिः गवाङ्ङ्भिः गवाङ्ङ्भिः 9
4	गोऽञ्चे गोअञ्चे गवाञ्चे	गोऽङ्भ्याम्म् गोऽङ्ङ्भ्याम् गोऽङ्ङ्भ्य्याम्म् गोअङ्भ्याम्म् गोअङ्ङ्भ्याम् गोअङ्ङ्भ्य्याम्म् गवाङ्भ्याम्म् गवाङ्ङ्भ्याम् गवाङ्ङ्भ्य्याम्म्	गोऽङ्भ्यः गोऽङ्ङ्भ्यः गोऽङ्भ्य्यः गोऽङ्भ्यः गोअङ्भ्यः गोअङ्ङ्भ्यः गोअङ्भ्य्यः गोअङ्भ्यः गवाङ्भ्यः गवाङ्ङ्भ्यः गवाङ्भ्य्यः गवाङ्भ्यः
5	गोऽच्चः गोऽञ्चः गोअच्चः गोअञ्चः गवाच्चः गवाञ्चः 6	गोऽङ्ङ्भ्याम्म् गोऽङ्भ्य्याम्म् गोअङ्ङ्भ्याम्म् गोअङ्भ्य्याम्म् गवाङ्ङ्भ्याम्म् गवाङ्भ्य्याम्म् 24	गोऽङ्ङ्भ्य्यः गोऽङ्ङ्भ्यः गोऽङ्भ्य्यः गोअङ्ङ्भ्य्यः गोअङ्ङ्भ्यः गोअङ्भ्य्यः गवाङ्ङ्भ्य्यः गवाङ्ङ्भ्यः गवाङ्भ्य्यः 24 गोऽङ्ङ्भ्य्यः गोअङ्ङ्भ्य्यः गवाङ्ङ्भ्य्यः
6	same as above 6	गोऽच्चोः गोऽञ्चोः गोऽञ्च्चोः गोअच्चोः गोअञ्चोः गोअञ्च्चोः गवाच्चोः गवाञ्चोः गवाञ्च्चोः 9	गोऽच्चाम्म् गोऽञ्चाम् गोऽञ्च्चाम्म् गोअच्चाम्म् गोअञ्चाम् गोअञ्च्चाम्म् गवाच्चाम्म् गवाञ्चाम् गवाञ्च्चाम्म् 9
7	गोऽच्चि गोऽञ्चि गोऽञ्च्चि गोअच्चि गोअञ्चि गोअञ्च्चि गवाच्चि गवाञ्चि गवाञ्च्चि 9	गोऽच्चोः गोऽञ्चोः गोऽञ्च्चोः गोअच्चोः गोअञ्चोः गोअञ्च्चोः गवाच्चोः गवाञ्चोः गवाञ्च्चोः 9	गोऽङ्क्षु गोऽङ्क्षु गोऽङ्ङ्क्षु गोऽङ्ङ्क्ष्षु गोऽङ्क्षु गोऽङ्क्ष्षु गोऽङ्ख्षु गोअङ्ङ्क्षु गोअङ्ङ्क्षु गोअङ्ङ्क्षु गोअङ्ङ्क्ष्षु गोअङ्क्षु
	गोऽङ्ङ्क्ष्षु गोऽङ्ङ्क्ष्षु गोऽङ्क्षु गोऽङ्ङ्क्ष्षु गवाङ्क्षु गोअङ्क्ष्षु गवाङ्ङ्क्ष्षु गवाङ्ङ्क्ष्षु गोऽङ्क्ष्षु गवाङ्ङ्क्षु गवाङ्ङ्क्षु गोऽङ्क्ष्षु गोऽङ्क्ष्षु गोअङ्ङ्क्ष्षु गोऽङ्क्ष्षु गवाङ्क्ष्षु गोअङ्क्ष्षु गवाङ्ङ्क्षु गवाङ्ङ्क्षु गवाङ्ङ्क्ष्षु गवाङ्क्ष्षु गवाङ्क्ष्षु गोअङ्ङ्क्ष्षु गोअङ्क्ष्षु गोअङ्ङ्क्षु गोअङ्ङ्क्षु गवाङ्क्ष्षु गवाङ्ङ्क्ष्षु गवाङ्क्ष्षु गवाङ्ङ्क्ष्षु गोअङ्क्ष्षु गोऽङ्ङ्क्षु गोऽङ्क्ष्षु गोऽङ्ङ्क्ष्ष्षु गोअङ्ङ्क्ष्षु गोअङ्ङ्क्ष्षु 51 गोअङ्ङ्क्ष्ष्षु गवाङ्ङ्क्ष्षु गवाङ्ङ्क्ष्ष्षु		

Modeling Pāṇinian Grammar

Peter M. Scharf
Brown University
25 June 2007

It is possible to achieve the implementation of generative grammars and parsers of Sanskrit using various methodologies which have varying degrees of affinity to those of Pāṇinian grammar. The current paper compares obvious methods to implement various aspects of Sanskrit grammar computationally, comments upon the degree to which they approach or depart from Pāṇinian methodology and exemplifies methods that would achieve a closer model.

I. Differences among the Sanskrit grammarians and even among Pāṇinians.

In attempting to create a computational model of Pāṇinian grammar, the first problem is to determine which Pāṇinian grammar. The *Aṣṭādhyāyī* itself (late 5th c. B.C.E.), consisting of nearly 4,000 rules, is known to have undergone modifications. Kātyāyana's approximately 4,300 vārtikas (4th-3rd c. B.C.E.) suggest modifications to 1,245 of Pāṇini's rules, usually in the form of additions (*upasaṅkhyāna*). Patañjali's *Mahābhāṣya* (mid-2nd c. B.C.E.) rejects many additions suggested by Kātyāyana, suggests other desiderata (*iṣṭi*), and articulates principles presupposed in the grammar. Many of the modifications Kātyāyana and Patañjali suggest are found adopted in the form in which the rules are found in Jayāditya and Vāmana's *Kāśikā*, the oldest extant complete running commentary on the *Aṣṭādhyāyī* (7th c. C.E.). Does one wish to model the *Aṣṭādhyāyī* alone? The *Aṣṭādhyāyī* and Kātyāyana's vārtikas? The grammar as known and approved by Patañjali in the *Mahābhāṣya*? Or the grammar as found in the *Kāśikā*?

II. Ambiguities in early articulations explicated differently by subsequent Indian linguists.

Articulations of Pāṇinian grammar, especially sūtras and vārtikas isolated from commentary, are subject to ambiguities. These ambiguities are resolved in different ways by different commentators. Commentaries on Patañjali's *Mahābhāṣya* disagree with each other; commentaries on the *Kāśikā* disagree with each other; and Bhaṭṭojidīkṣita's *Siddhāntakaumudī* (17th c. C.E.) differs in its interpretation of rules and procedures from Jayāditya and Vāmana's *Kāśikā*. Moreover, subcommentaries differ in their interpretations. Where ambiguities are found, how are they to be resolved? Using some particular commentator as the authority? haphazardly? Or is one going to come to an independent judgment of the correct interpretation after a critical evaluation of the various interpretations?

Moreover, the supplements to the grammar, particularly the lists referred to in various rules (*gaṇa*s), most prominently the list of roots, *Dhātupāṭha*, have undergone variation. Three complete commentaries composed in Sanskrit are extant on the Pāṇinian *Dhātupāṭha*, which is known only through these commentaries: the *Kṣīrataraṅginī* of Kṣīrasvāmin (early twelfth century C.E. Kashmir), the *Dhātupradīpa* of Maitreyarakṣita (mid-twelfth century C.E. Bengal), and the *Mādhavīyadhātuvṛtti* of Sāyaṇa (fourteenth century C.E. Vijayanagara, Karnataka). Will one use one of these? A unified critical edition of them? Or will one attempt to reconstruct the *Dhātupāṭha* as known to Patañjali?

Before embarking on a computational implementation of Pāṇinian grammar, such decisions ought to be made. It may prove very interesting to compare

computational implementations based upon different rule sets, different interpretations, and different sets of supplementary lists with each other and with different sets of linguistic data. As I have argued in two papers, with respect to the derivation of subjunctives (2005) and of the present stems of class eight roots (forthcoming), systematic comparison of linguistic descriptions resulting from computational implementations with each other and with various collections of extant Sanskrit texts may throw important light upon interpretational and historical questions.

III. Utilization of contemporary linguistic models, in particular those derived from Pāṇinian methodology, to articulate Pāṇinian methodology.

Indian grammatical commentaries composed in Sanskrit over the last two and half millennia are not the only sources of Pāṇinian interpretation. Recent work in theoretical and computational linguistics has influenced the interpretation of Pāṇinian grammar.

A. Influence of Pāṇinian methodology on contemporary linguistics generally.

Pāṇinian grammar has had a profound influence on modern linguistics. Apart from the influence of ancient Indian phonology on modern phonetic feature analysis, and the emulation of ancient Indian synchronic sound change laws by diachronic laws of phonological change in modern historical and comparative linguistics, Pāṇinian grammar is the archetype at the foundation of modern generative grammar. From Chomsky's first work on transformational grammar in 1957 to the Pāṇinian grammars of modern Indian languages such as described for Hindi in Bharati et al 1995, modern linguistic science has drawn heavily from the concepts and procedures of ancient Indian linguistics.

B. Influence of contemporary linguistic models on the interpretation of Pāṇinian methodology.

Concepts originally inspired by ancient Indian linguistics have taken their own shape in contemporary linguistics. They have responded to different concerns and been adapted to different questions. These new concepts have been applied by contemporary scholars to the interpretation of Pāṇinian grammar. One of the most prominent of these is the idea that grammar consists of modules in a generative hierarchy, or levels.

IV. Levels

A. Kiparsky's architecture

Clearly influenced by Chomskian generative grammar, Kiparsky and Staal (1969) proposed that Pāṇinian grammar contains rules in a hierarchy of four levels of representation: semantics, deep structure, surface structure, and phonology. More recently Kiparsky (2002) restates this scheme referring to the four levels as follows: semantic, morphosyntactic, abstract morphological, and phonological. Three classes of rules map prior levels onto subsequent levels: (1) rules that assign kārakas and abstract tense, (2) morphological spellout rules, and (3) rules of allomorphy and phonology. Rules incorporate conditions at both the levels from which and to which they map, as well as at prior levels in a unidirectional derivation beginning with semantics and ending with phonology.

B. Houben 1999

Houben (1999) aptly criticized earlier articulations of this four-level hierarchy because they did not explicitly include pragmatics and intentionality in the semantic level and did not permit semantic factors (including pragmatics and intentionality) to serve as conditions in phonological rules directly. In addition, he criticized the portrayal of Pāṇini's grammar as a complete automaton that produces utterances from meanings. He pointed out that there are no rules that introduce verbal roots and nominal stems based upon semantic

conditions and that the fundamental morphophonemic elements appear in Pāṇinian derivations from the start. It is therefore improper, he argued, to characterize the grammar as originating in semantics and culminating in phonological form. Rather, he (1999: 48) stated, it originates in meaning mixed with form and culminates in a perfected form.

C. The purpose of the science of language

Houben is correct to point out that it is not the function of the *Aṣṭādhyāyī* to teach semantics. The science of grammar does not teach the communication of meaning that is already known from ordinary usage; rather, it teaches correct usage in the conveyance of the desired meaning. In his very first vārtika commented upon at length by Patañjali in the *Paspaśāhnika*, Kātyāyana places the function of grammar in the context of what is already known from ordinary behavior. There is an established relation between words and their objects, which is known from ordinary usage, such that certain words are used to denote certain objects. The purpose of using speech forms is to convey knowledge of objects by following the conventions of ordinary usage. Since this is the case, the purpose served by the science of grammar is to make known which speech forms among those in use are correct and hence lead to dharma. Kātyāyana states:

> *Siddhe śabdārthasambandhe lokato 'rthaprayukte śabdaprayoge śāstreṇa dharmaniyamaḥ, yathā laukikavaidikeṣu.*[1]
> Since speech, its object, and the relation between the two are established (and are known) from ordinary usage, and since one uses speech prompted by meanings in accordance with ordinary usage, the science (of grammar) restricts (usage to correct speech forms) for the sake of dharma just as (other disciplines restrict behavior) in ordinary and Vedic affairs.[2]

D. Semantics

While it is ostensibly correct that the *Aṣṭādhyāyī* does not include any rules that are concerned with semantics to the exclusion of syntax, morphology, and phonology, the system of rules clearly presupposes that semantics drive the derivation. Meaning is the reason for speech. Under 1.1.44, Patañjali describes that the purpose of speech is to convey understanding:

> The use of words is for the purpose of the comprehension of the objects they denote. With the intention, "I will give the understanding of an object" a word is used.[3]

Modeling the fact that a speaker selects speech forms to use on the basis of the meaning he wishes to convey, the *Aṣṭādhyāyī* is composed in a manner that selects certain speech forms for use on the basis of certain semantic conditions. Specific semantic factors pervasively serve as conditions for the classification of lexical items, and for the introduction of kāraka terms, cover symbols, and speech forms.

1. Lexical organization

The use of words in rules to refer to classes of words rather than just to their own speech form is discussed in *Mahābhāṣya* under 1.1.68 *svaṁ rūpaṁ śabdasyāśabdasañjñā.* The word *vṛkṣa* 'tree', etc. in 2.4.12 *vibhāṣā vṛkṣamṛga...* refers to terms for species of trees.[4] The word *sva* 'property', etc. in 3.4.40 *sve puṣaḥ* refers to itself as well as to its synonyms,[5] while the word *rājan* in 2.4.23 *sabhā rājāmanuṣyapūrvā* refers to its synonyms but not to itself. Finally, the word *matsya* in 4.4.35 *pakṣimatsyamṛgānhanti* refers to itself as well as to terms for species of fish. The use of words in the grammar to refer to classes of words rather than to the speech form itself succeeds through the intermediary of the words meaning, against the norm in the grammar for words to refer just to their own form. By referring to their meaning, in the way words are ordinarily used, the meaning of

---

1 K1.6.8.
2 Scharf 1995.

3 *Arthagatyarthaḥ śabdaprayogaḥ. Arthaṁ saṁpratyāyayiṣyāmīti śabdaḥ prayujyate.* K1.105.2.
4 *sittadviśeṣāṇāṁ vṛkṣādyartham* vt 5, K1.176.25. The scheme of distinguishing the ways in which words are used to refer to various classes of words or to themselves proposed in vārtikas 5-8 is not adopted in the *Aṣṭādhyāyī.* It nevertheless illustrates these various usages in the grammatical treatise.
5 *pitparyāyavacanasya ca svādyartham.*

the word can serve as the condition to class groups of words of related meaning.

There are 735 words used in the locative to state semantic conditions in rules (including repetitions and excluding individual compound elements). Conditions that serve to classify lexical items include place (*deśa*),[6] district (*janapada*),[7] river (*nadī*),[8] mountain (*parvata*),[9] measure (*parimāṇa*),[10] genus,[11] species,[12] or ethnicity (*jāti*),[13] age (*vayas*),[14] fish (*matsya*), conscious being (*cittavat*).[15]

2. Semantic conditions for kārakas, cover symbols, and phonetics

It is well known that the terms *dhruva* 'fixed point', etc. in rules 1.4.24-55 *dhruvamapāye ’pādānam*, etc. serve as semantic conditions for the introduction of kāraka terms, and that terms such as *bhūta* 'past', *vartamāna* 'present', and *bhaviṣyat* 'future', used in the locative in 3.2.84 *bhūte*, 3.2.123 *vartamāne laṭ*, and 3.3.3 *bhaviṣyati gamyādayaḥ*, serve to introduce l-affixes. Houben (1999: 46) has illustrated the use of semantic and pragmatic factors as conditions for purely phonetic modifications in 8.2.82-108 *vākyasya ṭeḥ pluta udāttaḥ*, etc.

3. x-vacana

A number of rules explicitly use the term *vacana* 'denoting' to designate the semantic conditions that serve as the criteria to class together words that denote entities in major categories. Hence semantic conditions serve to form a class of words that denote entities other than substances (*asattvavacana*),[16] a class of words that denote qualities (*guṇavacana*),[17] a class of words that denote common properties (*sāmānyavacana*),[18] or distinguishing properties,[19] and a class of words that denote the essence (*bhāva*) of what is denoted by the stem after which certain affixes forming such words occur (*bhāvavacana*).[20]

Similarly, other rules explicitly use the term *vacana* to designate the semantic conditions that serve as the criteria to form narrower classes of lexemes subject to common operations. Hence in one rule semantic conditions serve to form classes of indeclinables that denote proximity (*samīpa*), flourishing (*samṛddhi*), lack of prosperity (*vyṛddhi*), absence of an object (*arthābhāva*), going beyond (*atyaya*), unsuitability for the moment (*asaṁprati*), the appearance of a sound or word (*śabdaprādurbhāva*), posteriority (*paścāt*), a meaning of *yathā*, sequence (*ānupūrvya*), simultaneity (*yaugapadya*), similarity (*sādṛśya*), success (*saṁpatti*), completeness (*sākalya*), end (*anta*), and senses denoted by nominal terminations and other affixes provided by rules 5.3.1-26 (*vibhakti*).[21] In other rules the term *vacana* designates classes of words that denote remembrance (*abhijñā*),[22] stages of bodily growth (*vayas*),[23] haste (*kṣipra*),[24] wish (*āśaṁsā*),[25] boundary

---

[6] *deśa* 3.3.78, 4.2.52, 4.2.67, 4.2.119, 5.2.105, 5.2.135, 6.3.98, 8.4.9; *adeśa* 8.4.24.
[7] 4.2.81, 4.2.124.
[8] 4.2.85.
[9] 4.3.91.
[10] 4.3.153, 5.2.39.
[11] *jāti* 4.1.161, 5.2.133; *ajāti* 5.4.37, 6.4.171.
[12] 6.3.103.
[13] 6.2.10.
[14] *vayas* 3.2.10, 4.1.20, 5.1.81, 5.2.130, 5.4.141, 6.2.95; *avayas* 5.1.84.
[15] 5.1.89.
[16] 2.3.33 *karaṇe ca stokālpakṛcchrakatipayasyāsattvavacanasya.*
[17] 2.1.30 *tṛtīyā tatkṛtārthena guṇavacanena*, 4.1.44 *voto guṇavacanāt*, 5.1.124 *guṇavacanabrāhmaṇādibhyaḥ karmaṇi ca*, 5.3.58 *ajādī guṇavacanādeva*, 6.2.24 *vispaṣṭādīni guṇavacaneṣu*, 8.1.12 *prakāre guṇavacanasya.*
[18] 3.4.5 *samuccaye sāmānyavacanasya*, 8.1.73 *nāmantrite samānādhikaraṇe sāmānyavacanam.*
[19] 8.1.74 *vibhāṣitaṁ viśeṣavacane bahuvacanam.*
[20] The term *bhāvavacana* occurs in three sūtras: 2.3.15 *tumarthācca bhāvavacanāt*, 2.3.54 *rujārthānāṁ bhāvavacanānāmajvareḥ*, 3.3.11 *bhāvavacanāśca*, and the term *bhāvakarmavacana* in one: 6.2.150 *ano bhāvakarmavacanaḥ.*
[21] 2.1.6 *avyayaṁ vibhaktisamīpasamṛddhivyṛddhyarthābhāvātyayāsaṁpratiśabdaprādurbhāvapaścādyathānupūrvyayaugapadyasādṛśyasampattisākalyāntavacaneṣu.*
[22] 3.2.112 *abhijñāvacane lṛṭ.*
[23] 3.2.129 *tācchīlyavayovacanaśaktiṣu cānaś*, 5.1.129 *prāṇabhṛjjātivayovacanodgātrādibhyo ’ñ*, 6.3.85 *jyotirjanapadarātrinābhināmagotrarūpasthānavarṇavayovacanabandhuṣu.*
[24] 3.3.133 *kṣipravacane lṛṭ.*
[25] 3.3.134 *āśaṁsāvacane liṅ.*

(*maryādā*),[26] imagination or supposition (*saṁbhāvana*),[27] fitness (*paryāpti*),[28] half (*sāmi*).[29] In commenting upon several of these rules, the *Kāśikā* notes that the term *vacana* is used to include synonyms of the word that precedes it in compound.[30]

Elsewhere the term *vacana* explicitly designates the semantic condition for a particular triplet of nominal terminations, secondary affix, or finished form (*nipātana*). Such semantic conditions include master (*īśvara*),[31] virgin (*apūrva*),[32] momentary (*ādyanta*),[33] particular sort or manner (*prakāra*),[34] and extolled (*prakṛta*).[35] and dependent (*tadadhīna*),[36] The term *vacana* also designates a broad class of semantic conditions that serve as conditions for the formation of tṛtīyātatpuruṣa compounds. These include additional significance such as praise or censure (*adhikārtha*).[37]

E. Ontology

In addition to various specific semantic factors that serve as conditions for the classification of lexical items, and for the introduction of kāraka terms, cover symbols, and speech forms, the *Aṣṭādhyāyī* incorporates certain ontological presuppositions. The grammar presupposes a certain structure in the semantic field in order to operate properly. Rules have been formulated with certain conceptions regarding the nature of things in mind. Numerous passages in Patañjali's *Mahābhāṣya* analyze such presuppositions as do the works of later philosophers of language from Bhartṛhari (5th century C.E.) to Kauṇḍabhaṭṭa and Nāgeśa (seventeenth and eighteenth centuries). Patañjali, for instance, has his interlocutors asks questions concerning the nature of action, time, and change in the course of their arguments about the formulation and scope of rules. They ask:

> What do you consider action to be when you say "The term *dhātu* doesn't apply to the roots *as* (class 2), *bhū* (class 1), and *vid* (class 4)."?
> *kāṁ punaḥ kriyāṁ bhavān matvāhāstibhavatividyatīnāṁ dhātusaṁjñā na prāpnotīti.* (cf. vt. 5)
> 1.3.1 K1.258.8-9)

> What do you consider time to be when you say "The object denoted by the word with which the word for time is compounded is not what gets measured, so the rule doesn't make sense."
> *kaṁ punaḥ kālaṁ matvā bhavān āha kālasya yena samāsas tasyāparimāṇitvād anirdeśa iti* (vt. 1). 2.2.5 K1.409.21-22)

> What do you consider change to be when you say, "It doesn't work (the taddhita suffix doesn't apply) in the case of *bali* and *ṛṣabha*."?
> *kaṁ punar bhavān vikāraṁ matvāha balyṛṣabhayor na sidhyati.*
> 5.1.13 K2.342.16)

Examination of the *Aṣṭādhyāyī* itself reveals that it presupposes a certain ontology. Substances (*dravya*), qualities (*guṇa*), and actions (*kriyā*) are distinguished as are time (*kāla*), the divisions of time past (*bhūta*), present (*vartamāna*), and future (*bhaviṣyat*), and the degrees of proximity in time near (*āsanna*), today (*adyatana*), and not today (*anadyatana*). Number (*saṁkhyā*) is recognized. Common properties (*sāmānya*) are recognized as is also essence (*bhāva*). Much of this ontology subsequently appears as categories in the Vaiśeṣeṣika system of philosophy.

The various ontological categories refered to in the *Aṣṭādhyāyī* serve as the

---

[26] 3.3.136 *bhaviṣyati maryādāvacane 'varasmin*, 8.1.15 *dvandvaṁ rahasyamaryādāvacanavyutkramaṇayajñapātraprayogābhivyaktiṣu.*

[27] 3.3.155 *vibhāṣā dhātau sambhāvanavacane 'yadi.*

[28] 3.4.66 *paryāptivacaneṣvalamartheṣu.*

[29] 5.4.5 *na sāmivacane.*

[30] Under 3.2.112, 3.3.133, 5.4.5 the *Kāśikā* states: *vacanagrahaṇaṁ paryāyārtham.*

[31] 2.3.9 *yasmādadhikaṁ yasya ceśvaravacanaṁ tatra saptamī.*

[32] 4.2.13 *kaumārāpūrvavacane.*

[33] 5.1.114 *ākālikaḍādyantavacane.*

[34] 5.3.23 *prakāravacane thāl*, 5.3.69 *prakāravacane jātīyar*, 5.4.3 *sthūlādibhyaḥ prakāravacane kan.*

[35] 5.4.21 *tatprakṛtavacane mayaṭ.*

[36] 5.4.54 *tadadhīnavacane.*

[37] 2.1.33 *kṛtyairadhikārthavacane.* The Kāśikā comments, "The expression of additional meaning is the expression of the superimposed meaning connected with praise or censure." (*stuti-nindā-prayuktam adhyāropitārtha-vacanam adhikārtha-vacanam*). In 2.3.46 *prātipadikārthaliṅgaparimāṇavacanamātre prathamā*, the term *vacana* is taken by commentators to denote number rather than to refer to reference explicitly, i.e. the rules does not provide as a condition for the occurrence of a first-triplet nominal termination merely the denotation (*vacana*) of measure (*parimāṇa*), gender (*liṅga*), and the meaning of the stem (*prātipadikārtha*)

conditions that characterize sets of speech forms. Speech forms are subject to various operations on the condition that they do or do not denote a certain entity in a certain ontological category. The semantic condition is frequently placed in the locative. For example, 5.4.11 *kimettiṅavyayaghād āmv adravyaprakarṣe* provides a suffix *ām* (*āmu*) to a stem ending in a comparative and superlative affix *tara* or *tama* on the condition that the excellence to be denoted is not located in a substance (*dravya*). Similarly, the speech forms in the list beginning with *ca* are termed *nipāta* if they do not denote a substance (*sattva*).[38] They are subsequently termed indeclinable (*avyaya*).[39] Other ontological categories that serve as semantic conditions in the locative include time (*kāla*),[40] and essence.[41]

F. Challenges to unidirectionality

Although the *Aṣṭādhyāyī* does not provide explicit rules exclusively regarding semantics, the fact that it does incorporate extensive organization of the semantic field is significant. It is particularly significant that the organization of the semantic field is carried out in part on the basis of reference to syntactic and morphological elements. Such elements are generally introduced subsequently to and on the basis of semantic conditions. Hence, the organization of the semantic field by reference to syntactic and morphological elements challenges the assertion that a hierarchy of levels is unidirectional.

1. x-arthe

In the level hierarchy articulated by Kiparsky (2002), Pāṇini employs elements at levels two and three to specify semantic criteria at level one. Twenty-five of the 735 words that specify semantic criteria employ the term *artha* 'meaning' in order to specify semantic conditions on level one on the basis of morphosyntactic elements at level two and morphological elements at level three.[42] In one case, an abstract morphological element on level two is employed to specify a semantic item on level one that serves as a semantic condition for another abstract morphological element at level two. 3.4.7 *liṅarthe leṭ* provides that in Vedic the abstract morphological element *leṭ* occurs in the meaning of the abstract morphological element *liṅ*. In this case, the rule that assigns abstract tense incorporates conditions only at the levels from which and to which it maps; it thereby accords with the general restriction that rules incorporate conditions only at the levels from which and to which they map.

The remaining 25 rules containing words ending in the term *artha* that specify semantic criteria violate the enunciated condition that rules incorporate conditions only at the levels from which and to which they map, as well as at prior levels in the unidirectional hierarchy beginning with semantics and ending with phonology. They incorporate conditions at level three that specify semantic criteria at level one, two levels prior in the unidirectional hierarchy. Two examples suffice to demonstrate the problem. 1.1.9 *īdūtau ca*

---

38 1.4.57 *cādayo 'sattve.*

39 1.1.37 *svarādinipātam avyayam.*

40 2.3.64, 5.3.15.

41 3.1.107, 3.3.18, 3.3.44, 3.3.75, 3.3.95, 3.3.98, 3.3.114, 3.4.69, 4.4.144, 6.2.25. As a Buddhist, it is natural for Jayāditya to avoid accepting essence at the meaning of the word *bhāva*. Jayāditya understands the root *bhū* to refer to generic action (*kriyāsāmānya*); hence he takes the term *bhāva* to refer to the generic action common to the meaning of any root. In the *Kāśikā* under 3.3.18 *bhāve*, he states *kriyāsāmānyavācī bhavatiḥ* following Patañjali's statement *kṛbhvastayaḥ kriyāsāmānyavācinaḥ* (K2.144.20, K2.47.24, etc.). Since the affixes provided under the heading of 3.3.18 occur after roots, which denote action, the *bhāvavacana* words refered to in 3.3.11 would denote generic action *kriyāsāmānya* even if the term *bhāva* did refer to essence; the common property in all action is the essence of action. A long tradition of comment on the meaning of the term *bhāva* determines that it denotes non-vibratory action (*aparispandamāna-kriyā*).

42 *saptamyarthe* 1.1.19, *caturthyarthe* 1.3.55, *tṛtīyārthe* 1.4.85, *mātrārthe* 2.1.9, *anyapadārthe* 2.1.21, *cārthe* 2.2.29, *caturthyarthe* 2.3.62, *liṅarthe* 3.4.7, *tumarthe* 3.4.9, *kṛtyārthe* 3.4.14, *matvarthe* 4.4.128, *dhātvarthe* 5.1.118, *vidhārthe* 5.3.42, *jīvikārthe* 5.3.99, *śakyārthe* 6.1.81, *tadarthe* 6.1.82, *nityārthe* 6.2.61, *atadarthe* 6.2.156, *atadarthe* 6.3.53, *īṣadarthe* 6.3.105, *anyadarthe* 6.4.60, *śakyārthe* 7.3.68, *upamārthe* 8.2.101, *kṛtvo'rthe* 8.3.43, *adhyarthe* 8.3.51.

saptamyarthe provides that the sounds *ī* and *ū* occurring in the meaning of the seventh vibhakti in the *Padapāṭha* are termed *pragr̥hya* and therefore do not undergo sandhi. The rule thereby specifies a semantic element, the meaning of the seventh vibhakti, at level one on the basis of items termed the *seventh vibhakti*, namely morphological elements *i os su*, at level three. The semantic condition in turn specifies a phonological trait, the absence of sandhi, at level four. Similarly, 3.4.8 *tumarthe sesenaseasen...* specifies several affixes that occur in the same meaning as the infinitival affix *tum*. The rule thereby employs a morphological element -*tum* at level three to characterize a set of semantic conditions at level one, which then conditions allomorphs -*se*, -*sen*, etc. at level four.

The first example supports the criticism of earlier versions of the levels theory already articulated by Houben (1999) that it did not permit semantic factors to serve as conditions in phonological rules directly. 1.1.9 provides just what was not permitted: the semantic condition consisting of the meaning of the seventh vibhakti inhibits sandhi. The present version of the levels theory accommodates this criticism by permitting rules to incorporate factors at any prior level in the hierarchy as conditions. An additional problem not previously articulated, however, plagues the present version of the levels theory: rules incorporate factors at subsequent levels of the hierarchy as conditions at prior levels.

It is not licit to dismiss the problem by claiming that the use of the term *artha* serves merely to state synonymy at levels two and three and does not involve mapping to prior the prior semantic level. As Houben (1999) has been pointed out, Pāṇini does not state rules that operate exclusively on the semantic level. Yet, as I have demonstrated above, Pāṇini does incorporate organization of the semantic level in his rules. The organization of the semantic level is achieved in part by reference to syntactic and morphological criteria. Since syntactic and morphological criteria serve to express the structure of the semantic level, subsequent levels of the hierarchy, including the morphological level which is two removed, serve as conditions for prior levels.

2. x-vacana

In two cases, the semantic condition that serves to characterize a set of speech forms is specified by reference to levels considered to be subsequent to the semantic level in the hierarchy of four levels proposed by Kiparsky and Staal. In 6.2.150 *ano bhāvakarmavacanaḥ*, the kāraka term *karman* designates a class of items that serve as the semantic conditions that characterize a set of speech forms. In accordance with this rule, a subsequent compound element (*uttarapada*) that meets three conditions has its final vowel high-toned. The three conditions are the following: 1. it ends in an affix of the form *ana*; 2. it denotes non-vibratory action (*bhāva*) or a direct object (*karman*); and 3. it is preceded by a compound element denoting a kāraka. The fact that a kāraka is referred to as the direct object of the root *vac* in the term *vacana* is significant. It indicates that Pāṇini considered kārakas to be denotable just as purely semantic conditions are denotable.

In 2.1.6, one of the semantic conditions that serves to characterize a class of indeclinables is itself characterized by morphological criteria. The rule provides that indeclinables that occur in a number of senses combine with subsequent elements to form avyayībhāva compounds. The senses specified include those denoted by nominal terminations and other affixes provided by rules 5.3.1-26 (*vibhakti*). Hence the morphemes that constitute vibhaktis serve to characterize the semantic conditions under which certain indeclinables are used. Morphological criteria therefore serve as the grounds for the organization of semantics which was considered a prior level in the hierarchy proposed by Kiparsky and Staal. Note that the adoption of cyclicity in the formation of the compounds in question does not escape the problem of counterdirectionality in the hierarchical ordering. Regardless of whether rules that generate compounds and their accentuation occur subsequent to rules that generate their compound elements,

the indeclinable that constitutes the prior element of the avyayībhāva compound must have access to the morphological level even before the question of its entering into a compound arises. Indeclinables are classed according to semantic criteria that are themselves specified by morphological units.

3. Avoidance of circularity

Although the seventh vibhakti arises subsequently to its semantic conditions, yet it can serve as the criterion to characterize its semantic conditions without resulting in circularity much in the way circularity is avoided by evoking the concept of a *bhāvinī saṁjñā*. The relationship that the saptamī vibhakti has to its meaning, as referred to by the word *artha* in the term *saptamyarthe* in 1.1.9 is not a contemporaneous one. The saptamī occurs subsequently in the hierachy of levels. The meaning of the saptamī are those for which the saptamī will subsequently occur, just as a *bhāvinī saṁjñā* is used for something that will arise in a subsequent derivational stage. Circularity is avoided because speech is nitya. The meaning of the seventh is at a prior level of derivation to the seventh terminations. One would have to run through the hierarchy to level 3 to get the 7th terminations in order to establish the semantic range of the meaning of the 7th at level 1. But because speech is nitya, and its relationship to meaning is established, the meaning of the seventh is known even before any particular derivational sequence is exhibited.

(To be expanded and polished.)

[1.1.45, K1.111.2 - 1.112.17]

[1.1.1 vt. 9, K1.40.18 - 1.41.4]

C. Kārakas

As early as 1964, R. Rocher criticized the characterization of kārakas as syntactic categories, instead arguing that they are semantic. Calling them syntactico-semantic, Cardona (1976: 215 -224) countered that it is suitable to consider kārakas as a level between the purely semantic level and the level at which nominal terminations are introduced (the morphological level) because the rules that introduce kāraka terms include both semantic and co-occurrence conditions.

It is certainly the case that co-occurrence conditions enter into *kāraka* classification rules, and therefore that the *kāraka* classification is an intermediate stage of derivation between that of semantic conditions and that of the introduction of nominal terminations. It is possible that such an intermediate stage serves merely the purpose of procedural economy and does not imply that *kāraka* classification constitutes a level in any psychological or structural sense. Pāṇini may conceive of just two levels: semantic (*artha*) and phonetic (*śabda*). *kārakas* are objects intended in certain relations; the level of intention is that of meaning, that is, the semantic level. One prominent seventeenth century philosopher of language seems to favor the conception of kārakas as semantic categories. Kauṇḍabhaṭṭa in the "Subarthanirṇaya" of his *Vaiyākaraṇabhūṣaṇasāra* speaks of basic meanings for *kārakas*. He describes the rules that do not mention syntactic conditions as circumscribing general semantic domains for them. Yet the fact that Pāṇini formulated rules categorizing certain semantic items under certain syntactic conditions in exception to these domains may capture the conception, held by speakers of the language, of such categories as natural groups. Whether this sort of conceptualization comprises a level between the semantic and the morphological, or whether all conceptualization by virtue of being conceptual is semantic, is a moot point from the point of view of Pāṇinian procedure. In Pāṇinian procedure, kāraka classification does occupy an intermediate stage between purely semantic conditions and the introduction of morphological elements.

While procedurally kāraka rules intervene between semantics and phonetics, they involve both semantics and co-occurrence conditions themselves and thereby include within them semantic and phonetic parameters. From a psychological or structural perspective, therefore, they constitute a mixture of levels rather then an intermediate level.

D. L-affixes

In their description of levels, Kiparsky and Staal place l-affixes at the same level as kārakas. Kiparsky (2002: 3) describes "Assignment of kārakas (Th-roles) and of abstract tense" as the function of the first set of rules mapping the semantic level to the morphosyntactic level. The treatment of l-affixes by Pāṇini, however, differs markedly from the treatment of kārakas. Kārakas are terms (*sañjñā*). Section 1.4 classifies semantic objects intended to be expressed by a speaker in relational categories by calling them by a kāraka term. Speech forms are subsequently introduced under the condition that an item designated by a kāraka term is to be denoted. L-affixes, in contrast, are introduced under semantic conditions, just as other affixes are, and then are replaced by morphological elements; they serve therefore as abstract morphological elements themselves rather than as morphosyntactic representations.[43] Kiparsky differentiates abstract morphological representation from morphosyntactic representation. Therefore, if l-affixes belong to abstract morphological representation and kārakas to morphosyntactic representation, it is incorrect to assert that they occupy the same level in Pāṇinian grammar.

The concept of levels in Pāṇinian grammar, and the hierarchy of four levels proposed by Kiparsky and Staal, was inspired by divisions that evolved in modern linguistics. It is anachronistic to read them into the *Aṣṭādhyāyī.* Kiparsky himself (2002: 2) hedges his attribution of levels to Pāṇini calling them, "what we (from a somewhat anachronistic modern perspective) could see as different levels of representation." Pāṇini's grammar certainly worked with two levels: meaning and speech. Its derivational procedure certainly included more than two stages. However, it appears forced to press the derivational stages into a conceptual hierarchy of levels between the purely semantic and the purely phonetic, particularly into a four-level hierarchy corresponding to modern linguistic divisions.

[43] Cardona (1997: 496) calls them "abstract affixes".

In describing Pāṇinian procedure, one must be clear about when one is superimposing conceptions from contemporary linguistics on Pāṇini. Likewise, in modeling Pāṇinian procedure one must be clear about when one is introducing contemporary computational procedures foreign to Pāṇini. In the next section, I describe the organization of Pāṇinian grammar, purely from a Pāṇinian perspective rather than from the perspective of modern theoretical linguistics. In the remainder of this paper, I differentiate computational implementations of Pāṇinian grammar that model Pāṇinian procedure from applications of generative computational techniques to Sanskrit.

V. Sketch of an overview of Pāṇinian architecture

The grammar is set up to derive correct speech forms from an open lexicon under certain conditions. The usual conditions are semantic, i.e. that certain meanings are to be denoted. Occasionally, conditions include pragmatics and literary context. In general, therefore, the grammar derives speech forms from meaning rather than vice versa. The grammar is not organized to determine the meaning of statements; it proceeds from the speakers point of view, not from the listeners point of view. It answer the question, "How do I say x?," not the question, "What does x mean?"

A. Introduction of basic elements on semantic conditions

In general Pāṇinian grammar introduces basic speech elements, or morphological elements, under semantic conditions. Basic speech elements include roots, nominal bases and affixes. Roots are introduced in two ways:

(1) Elements listed in the dhātupāṭha are termed roots (*dhātu*) by rule 1.3.1 *bhūvādayo dhātavaḥ.*

(2) Derived elements terminating in any of a series of affixes introduced in rules 3.1.5-31 are termed roots by rule 3.1.32.

Nominal bases are likewise introduced in two ways:

(1) Any meaningful element other than a root (*dhātu*), affix (*pratyaya*), or an element that terminates in an affix, whether listed or not, is termed a nominal base (*prātipadika*) by 1.2.45 *arthavad adhātur apratyayaḥ prātipadikam.*

(2) Derived elements, including both those terminating in affixes termed *kr̥t* or *taddhita* and compounds (*samāsa*), are termed nominal base by 1.2.46 *kr̥ttaddhitasamāsāś ca.*

Affixes are introduced by rules in adhyāyas 3-5 governed by the heading 3.1.1 *pratyayāḥ.* These include affixes in the list designated by 3.3.1 *uṇādayo bahulam.*

The basic speech elements of the grammar do not constitute a fully specified set of elements. First, lists are not specified as part of the ruleset; they are specified by commentators subsequently, which leaves open to doubt which items were intended to be included by the author of the rules himself. Second, the grammar includes recursive procedures. The derivational procedure permits not only the derivation of nominal bases from roots and other nominal bases and the derivation of words from roots and nominal bases but also permits the derivation of roots from roots, roots from nominal bases, roots from nominal words, and nominal bases from words.

Aside from lists being in doubt and the presence of recursive derivation of elements, the set of basic elements is an open set since what is classed as a nominal base includes any meaningful element outside of a specified set. 1.2.45 reads, "any meaningful element other than ... is a nominal base." Moreover, commentators call many of the lists of nominal bases merely exemplary (*ākr̥tigaṇa*) rather than complete. Finally, the fact that by 3.1.8-11 verbal roots are derived from an unspecified set of nominal words (*pada*), which are in turn derived from the open set of nominal bases, makes verbal roots an open set as well.

Now, nominal bases are explicitly stated to be meaningful, and affixes are introduced under semantic conditions. While no statement of the grammar introduces underived roots under semantic conditions and the *Dhātupāṭha* list did not originally include semantic designations for them, they are assumed to be meaningful elements from the outset. Roots and nominal bases enter the grammar as that after which affixes are provided under specified conditions, prevalently including semantic conditions.

VI. Sandhi

VII. Inflection

VIII. Feminine affixes

IX. Derivational morphology

A. nominal stems derived from roots

B. nominal stems derived from nominals (words *pada*)

taddhitas
compounds

C. secondary roots derived from roots

D. secondary roots derived from nominals

X. Metalanguage

See class notes from Pāṇinian grammar class, Filliozat's Intro.

A. Reference

1. phonetic elements, terms

1.1.68, etc.
anubandha designation 1.2.1-9
pratyāhāra formation 1.1.72? ādir antyena sahetā

1. Dhātupāṭha

Database of roots, markers
Rules segment the dhātupāṭha elements into morphemes and markers

2. Gaṇasūtras

3. Morphophonemic elements given in sūtras

affixes
kauṇḍinac

B. Rule structure

Rules of the Aṣṭādhyāyī determine the rule structure of sūtras:
1.1.49, 1.1.66-67 alo 'ntyasya, ādeḥ parasya, ādyantau ṭakitau, etc.

C. Rule ordering

paratva
nitya
antaraṅga-bahiraṅga
apavāda
asiddhatva
see Kiparsky p. 52

XI. Why model Pāṇini?

Attempting to work out details illuminates the understanding of Pāṇini's method.
Understanding Pāṇini's method better contributes to the improvement of linguistic methodology generally.
Working out models of Sanskrit generative grammar has direct benefits for Indological studies: bringing computational methods to the humanistic disciplines and bringing Indology into the digital humanities..

Bibliography

Bharati, Akshar, Vineet Chaitanya, and Rajeev Sangal. 1995. *Natural Language Processing: A Paninian Perspective.* New Delhi: Prentice-Hall of India.

Cardona, George. 1976. *Pāṇini: A Survey of Research.* The Hague: Mouton.

Chomsky, N. 1957. *Syntactic Structures.* The Hague: Mouton.

Houben, Jan. 1999 [2001]. "'Meaning statements' in Panini's grammar: on the purpose and context of the Astadhyayi." *Studien zur Indologie und Iranistik* 22: 23-54.

K *The Vyākaraṇa-mahābhāṣya of Patañjali.* Ed. Lorenz Franz Kielhorn. 3 vols. Third edition revised and furnished with additional readings references and select critical notes by K. V. Abhyankar. Pune: BORI, 1962, 1965, 1972. Reprint: 1985.

Kiparsky, Paul. 2002. "On the Architecture of Pāṇini's Grammar." Paul Kiparsky's Home Page. http://www.stanford.edu/~kiparsky/

Kiparsky, Paul and J. F. Staal. 1969. "Syntactic and semantic relations in Pāṇini." *FL* 5: 83-117.

Rocher, Rosane.1964. "'Agent' et 'objet' chez Pāṇini." *JOAS* 84: 44-54. p51.

Scharf, Peter. "Early Indian Grammarians on a speaker's intention," *Journal of the American Oriental Society* 115.1 (1995): 66-76.

Scharf, Peter. 2005. "Pāṇinian accounts of the Vedic subjunctive: leṭ *kṛṇvaíte.*" *Indo-Iranian Journal*, 48.1: 71-96. (Paper presented at the 214th Meeting of the American Oriental Society, 12-15 March 2004, San Diego, California.) [The publication is marred by publisher errors and omissions; correct version: http://www.language.brown.edu/Sanskrit/ScharfSubjunctive.pdf]

Scharf, Peter. forthcoming. "Pāṇinian accounts of the class eight presents." Paper presented at the 13th World Sanskrit Conference, 10-14 July 2006, Edinburgh.

SIMULATING THE PĀṆINIAN SYSTEM OF SANSKRIT GRAMMAR

Anand Mishra
Department of Computational Linguistics
Ruprecht Karls University, Heidelberg
`http://sanskrit.sai.uni-heidelberg.de`

ABSTRACT

We propose a model for the computer representation of the Pāṇinian system of sanskrit grammar. Based on this model, we render the grammatical data and simulate the rules of Aṣṭādhyāyī on computer. We then employ these rules for generation of morpho-syntactical components of the language. These generated components we store in a p-subsequential automata. This we use to develop a lexicon on Pāṇinian principles. We report briefly its implementation.

1. A REPRESENTATION OF Aṣṭādhyāyī

The general grammatical process of Aṣṭādhyāyī (Katre, 1989) can be viewed as consisting of the following three basic steps:

1. PRESCRIPTION of the fundamental components which constitute the language.
2. CHARACTERIZATION of these fundamental components by assigning them a number of attributes.
3. SPECIFICATION of grammatical operations based on the fundamental components and their attributes.

1.1. Fundamental Components

In his grammar Pāṇini furnishes a number of elements (phonemes/morphemes/lexemes) which constitute the building blocks of the language. We assign each of them a unique key in our database. Thus the phoneme /a/ has the key `a_0`, the *kṛt* suffix /a/ has the key `a_3` and the *taddhita* suffix /a/ is represented by the key `a_4`. Given such a collection of unique keys, we define the set of fundamental components as follows:

Definition 1 *The collection of unique keys corresponding to the basic constituents of the language, we define as the set $\mathcal{F}$ of fundamental components.*

Further, we decompose this set $\mathcal{F}$ into two disjoint sets $\mathcal{P}$ and $\mathcal{M}$, where $\mathcal{P}$ is the set of keys corresponding to the phonemes and $\mathcal{M}$ containing the keys of the rest of the constituting elements (morphemes/lexemes).

$$\mathcal{P} = \{\texttt{a_0}, \texttt{i_0}, \texttt{u_0}, \ldots\}$$
$$\mathcal{M} = \{\texttt{bhU_a}, \texttt{tip_0}, \texttt{laT_0}, \ldots\}$$

$$\mathcal{F} = \mathcal{P} \cup \mathcal{M} \tag{1}$$
$$\mathcal{P} \cap \mathcal{M} = \phi \tag{2}$$

1.2. Attributes

The fundamental units of the language are given an identity by assigning a number of attributes to them. The various technical terms introduced in the grammar come under this category, as also the *it* -markers and sigla or *pratyāhāra*s. For example, the attributes *hrasva*, *guṇa* and *a*C characterize the element `a_0` as short vowel /a/ and attributes like *pratyaya*, *prathama* and *ekavacana* tell that *ti*P (= `tip_0`) is a third-person singular suffix. Again, each attribute is assigned a unique key in our database.

Definition 2 *The collection of unique keys corresponding to the terms, which characterize a fundamental component, we define as the set $\mathcal{A}$ of attributes.*

Corresponding to the sets $\mathcal{P}$ and $\mathcal{M}$ we can decompose the set $\mathcal{A}$ into two disjoint sets $\mathcal{A}_\pi$ and $\mathcal{A}_\mu$, $\mathcal{A}_\pi$ being the set of unique keys of the attributes to the elements

of $\mathcal{P}$ and $\mathcal{A}_\mu$ to elements of $\mathcal{M}$.

$$\mathcal{A}_\pi = \{\texttt{hrasva_0}, \texttt{udAtta_0}, \texttt{it_0}, \ldots\}$$
$$\mathcal{A}_\mu = \{\texttt{dhAtu_0}, \texttt{pratyaya_0}, \texttt{zit_9}, \ldots\}$$

$$\mathcal{A} = \mathcal{A}_\pi \cup \mathcal{A}_\mu \tag{3}$$

We note that any two of the four sets $\mathcal{P}, \mathcal{M}, \mathcal{A}_\pi, \mathcal{A}_\mu$ are mutually disjoint.

2. BASIC DATA STRUCTURES

Given the set of fundamental components ($\mathcal{F} = \mathcal{P} \cup \mathcal{M}$) and the set of attributes ($\mathcal{A} = \mathcal{A}_\pi \cup \mathcal{A}_\mu$), we now define our data structure for representing the Pāṇinian process.

2.1. Sound Set ψ

Definition 3 *A sound set ψ is a collection of elements from sets $\mathcal{P}, \mathcal{M}$ and $\mathcal{A}$ having exactly one element from the set $\mathcal{P}$.*

$$\psi = \{\pi_p, \mu_i, \alpha_j | \pi_p \in \mathcal{P}, \mu_i \in \mathcal{M}, \alpha_j \in \mathcal{A}, i, j \geq 0\} \tag{4}$$

This is an abstract data structure. Although it corresponds to a phoneme or one sound unit, it represents more than just a phoneme.

2.2. Language Component λ

Definition 4 *A language component λ is an ordered collection of at least one or more sound sets.*

$$\lambda = [\psi_0, \psi_1, \psi_2, \ldots \psi_n] \textit{ such that } \|\lambda\| > 0 \tag{5}$$

CONVENTION: We use square brackets [ ] to represent an ordered collection and curly brackets { } for an unordered collection.

Language expressions at every level (phonemes, morphemes, lexemes, words, sentences) can now be represented as a language component.

Example 1: We represent the verbal root *bhū* as a language component λ.

$$\begin{aligned} \lambda &= [\psi_1, \psi_2] \text{ where} \\ \psi_1 &= \{bh, bh\bar{u}, dh\bar{a}tu, \ldots\} \\ \psi_2 &= \{u, bh\bar{u}, dh\bar{a}tu, ud\bar{a}tta, d\bar{\imath}rgha, a\mathrm{C}, \ldots\} \end{aligned}$$

Corresponding to the two phonemes *bh* and *ū* in *bhū*, we have an ordered collection of two sound sets ψ_1 and ψ_2. Consider the first one ψ_1: Its first element *bh* is from the phoneme set $\mathcal{P}$.[1] The second element *bhū* tells that the phoneme of this sound set is a part of the fundamental unit *bhū*. The third element stores the attribute *dhātu* (verbal root) to this sound set. Similarly, the second sound set ψ_2 has phoneme attributes which tell it to be an *udātta* (high pitched) *dīrgha* (long) *a*C (vowel).

Example 2: Similarly, the language component corresponding to the morpheme *l*AṬ is:

$$\begin{aligned} \lambda &= [\psi] \text{ where} \\ \psi &= \{l, l\text{AṬ}, pratyaya, ait, \d{t}it, \ldots\} \end{aligned}$$

Attribute *ṭit* says that it has *ṭ* as *it* - marker.

Example 3: The morphemes *bhū* followed by *l*AṬ can now be represented together by the language component

$$\begin{aligned} \lambda &= [\psi_1, \psi_2, \psi_3] \text{ where} \\ \psi_1 &= \{bh, bh\bar{u}, dh\bar{a}tu, \ldots\} \\ \psi_2 &= \{u, bh\bar{u}, dh\bar{a}tu, ud\bar{a}tta, d\bar{\imath}rgha, a\mathrm{C}, \ldots\} \\ \psi_3 &= \{l, l\text{AṬ}, pratyaya, ait, \d{t}it, \ldots\} \end{aligned}$$

Different linguistic units can now be identified by carrying out intersection with the appropriate subsets of $\mathcal{P}, \mathcal{M}$ and $\mathcal{A}$. For example to get the verbal root or *dhātu* in λ we take the intersection of an identity set ι = $\{dh\bar{a}tu\}$ with each of ψ_i's in λ and store the index i when the intersection-set is not empty. In this case we get the index list [1,2]. The list of ψ_i's corresponding to these indices then gives the searched morpheme. Thus, the verbal root is given by the language component [ψ_1, ψ_2].

2.3. Process Strip σ

Definition 5 *A process strip σ is an ordered collection of pairs, where the first element of the pair is the number of a particular grammatical rule (e.g. $rule_p$) and the second element is a language component λ.*

$$\sigma = [(rule_p, \lambda_p), (rule_q, \lambda_q), \ldots] \tag{6}$$

The rule number corresponds to the Aṣṭādhyāyī order and binds the process strip with a function implementing that rule in the actual program. Thus, the process

[1] Actually, it is the unique key bh_0 corresponding to the phoneme *bh* which is stored in the sound set.

strip simulates the Pāṇinian process by storing in the language component λ_p the effect of applying the rule $rule_p$.

3. BASIC OPERATIONS

Having defined our data-structure, we now introduce the basic operations on them.

3.1. Attribute Addition

Let $\alpha \subset \mathcal{A} \cup \mathcal{M}$ and ψ be a sound set. Then attribute addition is defined as

$$h_{a\psi}(\psi, \alpha) = \psi \cup \alpha \tag{7}$$

This operation can be applied to a number of sound sets given by indices $[i, i+1, \ldots, j]$ in a given language component λ

$$h_{a\lambda}(\lambda, \alpha, [i, \ldots, j]) = [\psi_1, \ldots, \psi_i \cup \alpha, \ldots, \psi_j \cup \alpha, \ldots, \psi_n] \tag{8}$$

Example 4: Consider the language component corresponding to the morpheme *Śa*P

$$\begin{aligned} \lambda &= [\psi] \text{ where} \\ \psi &= \{a, \text{Ś}a\text{P}, pratyaya, śit, pit, \ldots\} \end{aligned}$$

RULE tiṅ śit sārvadhātukam (3.4.113) says that affixes in the siglum *ti*Ṅ and those having *ś* as *it* marker are assigned the attribute *sārvadhātuka*. We implement this rule by checking if there are sound sets with attributes *pratyaya* together with *ti*Ṅ or *śit* and adding the attribute *sārvadhātuka* if the condition is fulfilled. In this case, we get:

$$\begin{aligned} \lambda &= [\psi] \text{ where} \\ \psi &= \{a, \text{Ś}a\text{P}, pratyaya, śit, pit, sārvadhātuka\} \end{aligned}$$

3.2. Augmentation

Let

$$\lambda = [\psi_1, \ldots, \psi_i, \psi_{i+1}, \ldots, \psi_n]$$
$$\lambda_k = [\psi_{1k}, \psi_{2k}, \psi_{3k}, \ldots, \psi_{mk}]$$

and i be an integer index such that $i \leq \|\lambda\|$, then augmentation of λ by λ_k at index i is defined as

$$h_g(\lambda, \lambda_k, i) = [\psi_1, \ldots, \psi_i, \psi_{1k}, \psi_{2k}, \psi_{3k}, \ldots, \psi_{mk}, \psi_{i+1}, \ldots, \psi_n] \tag{9}$$

Example 5: Consider the language component λ corresponding to the verbal root *bhū*.

$$\begin{aligned} \lambda &= [\psi_1, \psi_2] \text{ where} \\ \psi_1 &= \{bh, bhū, dhātu, \ldots\} \\ \psi_2 &= \{u, bhū, dhātu, udātta, dīrgha, a\text{C}, \ldots\} \end{aligned}$$

RULE vartamāne laṭ (3.2.123) says that the morpheme *l*AṬ is added after a *dhātu* if the present action is to be expressed. To implement this rule, we first look for the indices of sound sets which have the attribute *dhātu* and then append the sound set corresponding to *l*AṬ after the last index. We get,

$$\begin{aligned} \lambda &= [\psi_1, \psi_2, \psi_3] \text{ where} \\ \psi_1 &= \{bh, bhū, dhātu, \ldots\} \\ \psi_2 &= \{u, bhū, dhātu, udātta, dīrgha, a\text{C}, \ldots\} \\ \psi_3 &= \{l, l\text{AṬ}, pratyaya, ait, ṭit, \ldots\} \end{aligned}$$

3.3. Substitution

We define substitution in terms of the above two operations.

Let $[i, i+1, i+2, \ldots, j]$ be the indices of sound sets to be replaced in the language component $\lambda = [\psi_1, \ldots, \psi_i, \psi_{i+1}, \ldots, \psi_n]$.

Let $\lambda_k = [\psi_{1k}, \psi_{2k}, \psi_{3k}, \ldots, \psi_{mk}]$ be the replacement, then the substitution is defined as

$$h_s(\lambda, \lambda_k, [i, \ldots, j]) = h_g(h_{a\lambda}(\lambda, \{\delta\}, [i, \ldots, j]), \lambda_k, j) \tag{10}$$

where $\delta \in \mathcal{A}$ is the $attribute$ which says that this sound set is no more active and has been replaced by some other sound set.

Example 6: Consider the language component corresponding to the verbal root *ṇī*Ñ

$$\begin{aligned} \lambda &= [\psi_1, \psi_2] \text{ where} \\ \psi_1 &= \{ṇ, ṇī\text{Ñ}, dhātu, ñit\} \\ \psi_2 &= \{i, ṇī\text{Ñ}, dhātu, ñit, dīrgha, a\text{C}\} \end{aligned}$$

RULE ṇaḥ naḥ (6.1.065) says that the initial retroflex *ṇ* of a *dhātu* is replaced by dental *n*. To implement this rule we first search the sound sets corresponding to *dhātu*, check whether the first one has a retroflex *ṇ* and if the conditions are fulfilled, add the attribute δ in that sound set and append the sound set corresponding

to *n* after it. Further we transfer all attributes (except the phoneme attributes) from the *ṇ* - sound set to *n* - sound set for *sthānivadbhāva*. We get,

$$\begin{aligned}\lambda &= [\psi_1, \psi_2, \psi_3] \text{ where}\\ \psi_1 &= \{ṇ, ṇı\tilde{\text{N}}, \textit{dhātu}, \textit{ñit}, \delta\}\\ \psi_2 &= \{n, ṇı\tilde{\text{N}}, \textit{dhātu}, \textit{ñit}\}\\ \psi_3 &= \{i, ṇı\tilde{\text{N}}, \textit{dhātu}, \textit{ñit}, \textit{dīrgha}, a\text{C}\}\end{aligned}$$

4. GRAMMATICAL PROCESS

4.1. Representing a Rule of Grammar

We represent a rule of grammar through a function f_q, which takes a process strip σ_p and adds a new pair $(rule_q, \lambda_q)$ to it where $rule_q$ is the number of the present rule and λ_q is the new modified language component after application of one or more of the three operations defined above on the input language component λ_p.

$$\begin{aligned}f_q(\sigma_p) &= \sigma_q \text{ where}\\ \sigma_p &= [\ldots, (rule_p, \lambda_p)]\\ \sigma_q &= [\ldots, (rule_p, \lambda_p), (rule_q, \lambda_q)]\\ \lambda_q &= h_a, h_g, h_s(\lambda_p, \ldots)\end{aligned}$$

4.2. Structure of a rule

The general structure of a rule is as follows:

```
Function f_q with input strip:
σ_p = [..., (rule_p, λ_p)]
```

```
check applicability conditions
if conditions not fulfilled then
   return unextended σ_p
else
  create new modified λ_q
  return extended σ_q
```

Thus, given a particular state (represented by σ_p) in the process of generation, the system provides for checking the applicability of a rule f_q, and if the conditions are fulfilled, the rule is applied and the changed language component together with the rule number is stored in the modified state (represented by σ_q).

As the rule numbers are also stored, we can implement the rules of *tripādī* and make their effects invisible for subsequent applications. The order in which rules are applied is provided manually through templates.

5. EXAMPLE

We take a verbal root *bhū* and generate the final word *bhavati* meaning "he/she/it becomes". We initialize the process strip σ_0 by loading the language component corresponding to the verbal root and adding $a00000$ as the rule number.

$$\begin{aligned}\sigma_0 &= [(a00000, \lambda_0)] \text{ where}\\ \lambda_0 &= [\psi_{0a}, \psi_{0b}] \text{ with}\\ \psi_{0a} &= \{\textit{bh}, \textit{bhū}, \textit{dhātu}\}\\ \psi_{0b} &= \{u, \textit{bhū}, \textit{dhātu}, \textit{dīrgha}, \textit{udātta}\}\end{aligned}$$

RULE vartamāne laṭ (3.2.123) says that the morpheme *l*AṬ is added after a *dhātu* if the present action is to be expressed. The application now involves following steps: Look in the last language component λ of the process-strip σ. If there are sound sets ψ with the identity set ι = { *dhātu* } in it, get their indices in index list. This returns the index list [1,2]. If index list is non empty then augment the language component λ_0 by attaching the language component corresponding to the morpheme *l*AṬ. This is attached in this case at index 2 as the new morpheme comes after *dhātu*. Extend the process strip σ_0 accordingly.

$$f_{a32123}(\sigma_0) = \sigma_1$$

$$\begin{aligned}\sigma_1 &= [(a00000, \lambda_0), (a32123, \lambda_1)] \text{ where}\\ \lambda_1 &= [\psi_{0a}, \psi_{0b}, \psi_{1a}] \text{ with}\\ \psi_{0a} &= \{\textit{bh}, \textit{bhū}, \textit{dhātu}\}\\ \psi_{0b} &= \{u, \textit{bhū}, \textit{dhātu}, \textit{dīrgha}, \textit{udātta}\}\\ \psi_{1a} &= \{l, l\text{AṬ}, \textit{pratyaya}, \textit{ait}, \textit{ṭit}\}\end{aligned}$$

RULE tip tas jhi sip thas tha mip vas mas ta ātām jha thās āthām dhvam iṭ vahi mahiṅ (3.4.078) provides for substitution of *l*AṬ. We take the first suffix *ti*P for replacement. The sound sets to be replaced are determined by taking intersection with the set { *l*AṬ, *l*IṬ, *l*OṬ, . . . } which has the morphemes having cover term *l*. In this case it is at the index 3. We replace this sound set with *ti*P i.e. add the attribute δ to the sound set at index 3 and augment the language component at this index.

$$f_{a34078}(\sigma_1) = \sigma_2$$

$$\begin{aligned}
\sigma_2 &= [\ldots,(a32123,\lambda_1),(a34078,\lambda_2)]\\
\lambda_2 &= [\psi_{0a},\psi_{0b},\psi_{1a},\psi_{2a},\psi_{2b}]\\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu\}\\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, d\bar{\imath}rgha, ud\bar{a}tta\}\\
\psi_{1a} &= \{l, l\text{AṬ}, pratyaya, ait, \d{t}it, \delta, \}\\
\psi_{2a} &= \{t, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}\\
\psi_{2b} &= \{i, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

RULE kartari śap (3.1.068) says that the morpheme *ś*aP is added after *dhātu* but before *sārvadhātuka* suffix and denotes agent. Check if sound set with *sārvadhātuka* follows one with *dhātu*. If yes then augment the language component for *ś*aP after *dhātu*.

$$f_{a31068}(\sigma_2) = \sigma_3$$

$$\begin{aligned}
\sigma_3 &= [\ldots,(a34078,\lambda_2),(a31068,\lambda_3)]\\
\lambda_3 &= [\psi_{0a},\psi_{0b},\psi_{3a},\psi_{1a},\psi_{2a},\psi_{2b}]\\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu\}\\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, d\bar{\imath}rgha, ud\bar{a}tta\}\\
\psi_{3a} &= \{a, \text{ś}a\text{P}, pratyaya, hrasva, \acute{s}it, pit\}\\
\psi_{1a} &= \{l, l\text{AṬ}, pratyaya, ait, \d{t}it, \delta\}\\
\psi_{2a} &= \{t, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}\\
\psi_{2b} &= \{i, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

RULE yasmāt pratyaya vidhis tad ādi pratyaye aṅgam (1.4.013) makes the part before the suffix *ś*aP an *aṅga* with respect to it.

$$f_{a14013}(\sigma_3) = \sigma_4$$

$$\begin{aligned}
\sigma_4 &= [\ldots,(a31068,\lambda_3),(a14013,\lambda_4)]\\
\lambda_4 &= [\psi_{0a},\psi_{0b},\psi_{3a},\psi_{1a},\psi_{2a},\psi_{2b}]\\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga\}\\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, d\bar{\imath}rgha, ud\bar{a}tta\}\\
\psi_{3a} &= \{a, \text{ś}a\text{P}, pratyaya, hrasva, \acute{s}it, pit\}\\
\psi_{1a} &= \{l, l\text{AṬ}, pratyaya, ait, \d{t}it, \delta\}\\
\psi_{2a} &= \{t, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}\\
\psi_{2b} &= \{i, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

RULE sārvadhātuka ārdhadhātukayoḥ (7.3.084) says that before *sārvadhātuka* or *ārdhadhātuka* replace the *i*K vowels by *guṇa* vowels. As *ś*aP is *sārvadhātuka*, we get

$$f_{a73084}(\sigma_4) = \sigma_5$$

$$\begin{aligned}
\sigma_5 &= [\ldots,(a14013,\lambda_4),(a73084,\lambda_5)]\\
\lambda_5 &= [\psi_{0a},\psi_{0b},\psi_{5a},\psi_{3a},\psi_{1a},\psi_{2a},\psi_{2b}]\\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga\}\\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, d\bar{\imath}rgha, ud\bar{a}tta, \delta\}\\
\psi_{5a} &= \{o, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga\}\\
\psi_{3a} &= \{a, \text{ś}a\text{P}, pratyaya, hrasva, \acute{s}it, pit\}\\
\psi_{1a} &= \{l, l\text{AṬ}, pratyaya, ait, \d{t}it, \delta\}\\
\psi_{2a} &= \{t, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}\\
\psi_{2b} &= \{i, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

RULE ec aḥ ay av āy āv aḥ (6.1.078) says that before *a*C (vowel) *e*, *o*, *ai*, *au* are respectively replaced by *ay*, *av*, *āy*, *āv*.

$$f_{a61078}(\sigma_5) = \sigma_6$$

$$\begin{aligned}
\sigma_6 &= [\ldots,(a73084,\lambda_5),(a61078,\lambda_6)]\\
\lambda_6 &= [\psi_{0a},\psi_{0b},\psi_{5a},\psi_{6a},\psi_{6b},\psi_{3a},\psi_{1a},\psi_{2a},\psi_{2b}]\\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tua\dot{n}ga\}\\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, d\bar{\imath}rgha, ud\bar{a}tta, \delta\}\\
\psi_{5a} &= \{o, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, \delta\}\\
\psi_{6a} &= \{a, av, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, hrasva\}\\
\psi_{6b} &= \{v, av, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga\}\\
\psi_{3a} &= \{a, \text{ś}a\text{P}, pratyaya, hrasva, \acute{s}it, pit\}\\
\psi_{1a} &= \{l, l\text{AṬ}, pratyaya, ait, \d{t}it, \delta\}\\
\psi_{2a} &= \{t, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}\\
\psi_{2b} &= \{i, ti\text{P}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit\}
\end{aligned}$$

Finally we collect all ψ_is not having a δ, i.e. which are not already replaced. This gives us the desired form *bhavati*.

6. **PASSIM** (PāṇiniAN SANSKRIT SIMULATOR)

In the following we give a brief description of `PaSSim` (Pāṇinian Sanskrit Simulator) we are developing at the University of Heidelberg.[2] The program aims towards developing a lexicon on Pāṇinian principles. The user

[2] `http://sanskrit.sai.uni-heidelberg.de`

enters an inflected word or *pada* and the system furnishes a detailed, step by step process of its generation. It is written in Python™ and consists of the following modules (See Figure 1):

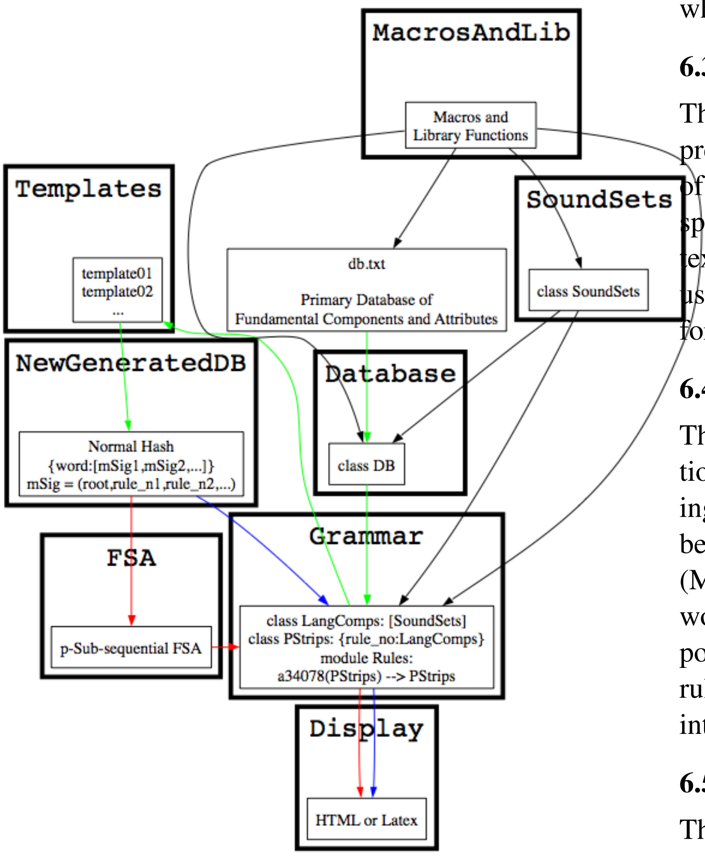


Figure 1: `PaSSim` (Pāṇinian Sanskrit Simulator)

6.1. `Database`

This module is for inputting, updating, enhancing and organizing the primary database of fundamental components and attributes. The organization of database serves the purpose of incorporating static information of Pāṇinian formulations. For example, *u*Ṅ is stored with *static* attributes *dhātu*, *bhvādi*, *aniṭ* and that its second phoneme is *it* - marker etc. Thus, the effect of many definition rules of Aṣṭādhyāyī are stored in the database. The database is in ASCII and each fundamental component or attribute has a unique key corresponding to which is a hash.

6.2. `Grammar`

This is the main module. It contains abstract classes corresponding to `SoundSets`, `LanguageComponents` and `ProcessStrips`. Further it has a number of functions like `a61065()`, which simulate the individual rules of Aṣṭādhyāyī.

6.3. `Templates`

This module is to organize the *prakriyā*. A `template` prescribes the rules in order of applicability for a group of primary verbs or nominal stems. Templates are specified manually, taking into account the *prakriyā* texts e.g. Siddhānta-Kaumudī (Vasu, 1905).[3] It uses `Grammar` to generate the morpho-syntactic word forms or *pada*s.

6.4. `FSA`

This module is for the sake of effecient representation of generated words together with the initializing fundamental component(s) and list of rule numbers. These are stored as a p-subsequential transducer (Mohri, 1996).[4] The output string associated with a word, thus provides the initializing fundamental components and a list of rules. `Grammar` applies these rules one after another and outputs the final as well as intermediate results.

6.5. `Display`

This module provides HTML[5] / LaTeX output. It outputs the content according to the given style sheet for conventions regarding script, color-scheme etc. The phonological, morphological, syntactical and semantical information gathered during the process of generation is rendered in modern terms through a mapping of Pāṇinian attributes corresponding to it.

7. REFERENCES

Böhtlingk, Otto von. 1887. *Pāṇini's Grammatik*. Olms, Hildesheim. Primary source text for our database.

[3]We would like to acknowledge two texts in Hindi — Vyākaraṇacandrodaya (Śāstrī, 1971) and Aṣṭādhyāyī sahajabodha (Dīkṣita, 2006) — which have been very beneficial to us.

[4]Sequential transducers can be extended to allow a single additional output string (subsequential transducers) or a finite number p of output strings (p - subsequential transducers) at final states. These allow one to deal with the ambiguities in natural language processing (Mohri, 1996).

[5]See: http://sanskrit.sai.uni-heidelberg.de

Dīkṣita, Puṣpā. 2006-07. *Aṣṭādhyāyī sahajabodha.* Vols. 1-4. Pratibha Prakashan, Delhi, India.

Katre, Sumitra M. 1989. *Aṣṭādhyāyī of Pāṇini.* Motilal Banarsidass, Delhi, India.

Mohri, Mehryar. 1996. On some Applications of Finite-State Automata Theory to Natural Language Processing. *Journal of Natural Language Engineering*, 2:1-20.

Śāstrī, Cārudeva. 1971. *Vyākaraṇacandrodaya.* Vols. 1-5. Motilal Banarsidass, Delhi, India.

Vasu, Srisa Chandra and Vasu, Vaman Dasa. 1905. *The Siddhānta-Kaumudī of Bhaṭṭojī Dīkṣita.* Vols. 1-3. Panini Office, Bhuvanesvara Asrama, Allahabad, India. Primary source text for *prakriyā*.

AN EFFORT TO DEVELOP A TAGGED LEXICAL RESOURCE FOR SANSKRIT

S. Varakhedi
V.Jaddipal
V. Sheeba
Rashtriya Sanskrit Vidyapeetha Deemed University
Tirupati
{shrivara,v.jaddipal,v.sheeba}@gmail.com

1. ABSTRACT

In this paper we present our efforts the first time of its kind in the history of Sanskrit to design and develop a structured electronic lexical Resource by tagging a Traditional Sanskrit dictionary. We narrate how the whole unstructured raw text of Vaacaspatyam – an encyclopedic type of Sanskrit Dictionary has been tagged to form a user friendly e-lexicon with structured and segregated information through corpus designing methods.

2. INTRODUCTION

It is not unknown to the scholars in the field of computational linguistics that electronic lexical resources are useful not only for human understanding but also for the needs of language processing. Many NLP applications like Morphological Analyzer etc. inevitably require a well-formed lexical resource. Lexical resource with grammatical and semantic information would be helpful in processing and in translation and decision making inference engines as well. It is not possible to do any kind of language processing in syntactic and semantic level without the structured relevant information regarding the stems of that language. Keeping this in view we have developed this e-resource for the Sanskrit language.

The Sanskrit language is one of the oldest classical languages of the world. It has a gigantic literary treasure related to all branches of sciences and all walks of life. Sanskrit is the first language to have a very precise grammar formalism authored by Paa.nini two thousand years ago. No other language has such a great tradition of grammar formalism, which is sound, perfect and very formal in nature. For these reasons Sanskrit gives enormous scope for NLP researchers and computer scientists from computational view point. With out proper lexical resources, NLP researchers will find themselves handicapped. This is the meeting point of traditional linguists and computational researchers. The information available in conventional lexicons are not sufficient for computational analysis. The way how information is stored becomes more crucial rather than how much information is available in the lexicon. Therefore the lexicographer of an electronic lexicon should be careful while designing the lexicon for computational processing purpose. In case of Sanskrit the design of e-lexicon is more complex because the traditionally available lexicons or dictionaries in Sanskrit have by nature structural complexities. They are designed in an organized but not so well organized for computational purposes. Nevertheless they become important for they carry tremendous and immeasurable information. Hence restructuring of such available dictionaries in Sanskrit is the pre-eminent necessity of Sanskrit computational linguistics. In this direction, our team has opted Vaacaspatyam for tagging on an experiment basis with a goal of developing a multipurpose electronic lexical resource that could be useful for academic research and computational processing as well. This work opens up further a new avenue of research in development of Sanskrit electronic dictionaries in a similar method. The experience gained through this program has prompted us to take up V.S.Apte Sanskrit-English Dictionary in hand.

3. HARD-BOOK TO SOFT-BOOK

This encyclopedic type of lexicon was first published in Kolkata in 1884. It is needless to say that the re-printed editions that are available now are not at all in readable condition due to old font types, unclear print and missing characters. Apart from all this,

untraced errors in the original print often mislead the readers. It was felt necessary to have an electronic version of this gigantic work which runs into about 11 thousand pages in six big volumes printed in old kolkata printing halls using small fonts. There are no breaks in words, not even clear breaks in topics and paragraphs. For each entry tremendous information is collected. Nevertheless, everything is undivided and hence not easily accessible even by eminent scholars. To avoid all this trouble, we initiated to develop an e-content of the Vaacaspatyam, which can be searched and retrieved with out any hustle. We started keying in the data into machines in ISCII using gist technology developed by CDAC Pune. Within a year we got the first raw version of the original text that needed several readings to get the proof corrected.

4. INTRODUCTION TO THE VAACASPATYAM

Pandit **Taaranaatha Tarkavaacaspati**, with his in-depth erudition and indefatigable industriousness devoted several years to prepare encyclopedic Sanskrit Lexicon called ‘Vaacaspatyam’, consisting of about 5442 printed pages of A4 size. It contains terms along with their derivations and explanations drawn from almost all the branches of Sanskrit Literature, such as the Vedaas, Vedaa”ngaas, Puraa.naas, Upapuraa.naas, Philosophy, Tantra, Artha”saastra, Ala”nkaara”saastra, Chan.da”s”saastra, Sangitasastra, Military Science, Paakavi.dyaa, “Sik.sa, Kalpa, Hasti “Saastram, Ha.tha-Yoga and Vaastu”saastra etc. Besides these, the technical words and doctrines of the following systems of Philosophy are fully explained: Caarvaaka, Maadhyamika, Yogaacaara, Vaibha.sika, Soutraantika, Arhata, Raamaanuja, Maadhva, Paa”supata, “Saiva, Pratyabhij˜na, Rase”swara, Paa.niniiya Vyaakara.na, Nyaaya, Vai”se.sika, Mimaamsa, Saa”nkhya, Pata˜njali-Yoga “Sastra and Vedaanta.

It is needless to say that lexicon is an essential part of language -learning. The most ancient language of the world, Sanskrit, also has had Lexicons such as Amarakosa, Vaijayanti, Vi”swaroopako”sa, Naanaarthako”sa etc., which serve more like Thesaurus than dictionary. However, dictionaries such as “Sabdakalpadruma of Raadhaakaantadeva were compiled, where in the grammatical requirements of the Sanskrit reader were also met with. The Vaacaspatyam as a lexicon of Sanskrit words with word-derivation, grammatical specification as per the Paa.ninian System is an excellent work for reference concerning the Sanskrit system of word-meaning. The uniqueness of this lexicon, in comparison to even its succeeding lexicons such as Apte's dictionary, is that while most dictionaries concentrate on the semantic aspect of words listed, the Vaacaspatyam not only deals with their grammatical aspects but also gives all the details available in Sanskrit literature. Though the Vaacaspatyam is constructed in the style of “Sabdakalpadruma, it excels “Sabdakalpadruma in references and size.

This Sanskrit lexicon which is encyclopedic in nature is a pioneering work of its kind and ever since its publication has been held in the highest esteem not only in India but even in England and Europe, because it is by far, wider and deeper in scope than any other contemporary Sanskrit dictionary.

The author of the work Pandit Taaraanatha Tarkavaacaspati himself states that in addition to all the derivations and different meanings with illustrations of all the words which are found in Wilson's Sanskrit Dictionary and Raja Raadhaakanta's “Sabdakalpadruma, Vaacaspatyam contains numerous Vedic words which are not found even in the Bohtlink's St. Petersburg Sanskrit - German dictionary.

5. CONTENTS AND FEATURES OF VACHASPATYAM

1. Listing of words in alphabetical order.
2. Paa.nini’s li”ngaanu”saasana on genders.
3. Panini’s rules on the suffixes.
4. Paa.nini’s rules on the primitive and derivative words.
5. The derivation and different meanings with illustrations of all the words which are found in the Wilson's Sanskrit dictionary and Raadhaakaanta's “Sabdakalpadruma and numerous words not to be found in the said or any previous dictionaries.
6. The derivations and different meanings of the words of the Vedas.
7. Numerous Vedic words not to be found in Bohtlink's Sanskrit and German dictio-

nary.Technical words and doctrines of the following system of Philosophy are fully explained - Caarvaaka, Maadhyamika, Yogacara, Vaibha.sika , Soutraantika, Aarhata, Raamaanuja, Maadhva, Paa"supata, "Saiva, Pratyabhij˜na, Rase"swara, Paa.nini, Nyaaya, Vai"se"sika, Miimaamsa, Saankhya, Paatanjala-Yoga and Vedaanta.

8. The technical terms of the "Srouta and G.rhya sutras.

9. The technical words of Sm.ritis.

10. The plan and scope of all the Puraa.naas and Upapuraa.naas.

11. Plan and scope, of the Mahaabhaarata and Raamaaya.na.

12. The History of the ancient Kings of India as far as gathered from the Puraa.naas and Upapuraa.nas.

13. The position of the different countries/dwiipaas according to ancient Indian Texts. (Brahmaan.davar.nanam)

14. The full explanation of technical terms of Ayurveda and also an account of Ancient Indian Science of Anatomy and the preparation of the medicines according to Ayurvedic texts.

6. DEVELOPMENT OF E-VAACASPATYAM

The Vaacaspatyam contains about 46970 unique word entries with explanation. Each entry has a minimum of 2 lines of information and in most of the cases it runs about 10 to 20 lines. Some prominent words may have elaborate entries upto 20 pages on their category, meaning, sources, usages and other related information. More than 200 source books of different branches and disciplines of learning and are cited. References of more than 30 ko"shaas (lexicons) are found. Names of these references and sources that were cited in short abbreviations are expanded to their full form for the benefit of the readers. The tags help in segregating such flowing information.

7. TAGGING SCHEME FOR E-VAACASPATYAM

As soon as the basic electronic text was ready for application, we started to tag the text with meta-tags to identify the structures of the lexicon. The following tagset was developed for marking.

Tag Description Example

1. <cat> Category of the word a <cat> avyaya </cat>

2. <vp> Vyutpatti i.e., etymology aja <vp> na jaayate </vp>

3. <pr> PrayogaH – usage in any standard Sanskrit work

4. <vkr> Vyakarana information aja<vkr> na+janii+da </vkr>

5. <ar> Artha i.e., meaning aja <ar> caturmuKa </ar>

6. <vg> Vigraha i.e, explanation given for compound word

7. <akr> Aakara i.e., source for the word or its etymology, usage etc.

8 <vn> Vivara.nam i.e., narration about the particular concept

This tag set is used to segregate the semantic part of the text. For stylistic presentation, we have used some other tags (<sl> to indicate "Sloka etc.) which are not listed here. Initially the tagging was done manually with help of some scholars. Later, we could find out some heuristics using which 70% of the text was mechanically tagged. Through this method we saved human labor as well as money.

The raw text added with these meta-tags, which contain necessary information about linguistic features, has become a good resource not only for presentation but also for better understanding of the original text and the semantics of the words listed in the lexicon.

This Tagging scheme has given tree structure to the lexicon in the following way.

A) Simple structure 1

<word> %stem%
<category> INFO </category>
<grammar> INFO </grammar>
<meaning1> INFO </meaning1>
<usage1> INFO </usage1> . . . ..
<ref1> INFO </ref1> . . . .
<meaning2> INFO </meaning2> . . . .
</word>

B) Structure 2

<word> %stem%
<grammar> INFO </grammar>
<category1> INFO </category1>
<meaning1> INFO </meaning1>
<usage1> INFO </usage1> . . . ..

<ref1> INFO </ref1>. . . .
<usage2> INFO </usage2>
<meaning2> INFO </meaning2> . . . .
<category2> INFO </catgory2>
<meaning1> INFO </meaning1>
<usage1> INFO </usage1> . . . ..
<ref1> INFO </ref1> . . .
<usage2> INFO </usage2>
<meaning2> INFO </meaning2> . . . .
</word>
C) Structure 3
<word> %stem%
<category> INFO </category>
<grammar1> INFO </grammar1>
<meaning1> INFO </meaning1>
<usage1> INFO </usage1> . . . ..
<ref1> INFO </ref1>. . . .
<usage2> INFO </usage2>
<meaning2> INFO </meaning2> . . . .
<grammar2> INFO </grammar2>
<meaning1> INFO </meaning1>
<usage1> INFO </usage1> . . . ..
<ref1> INFO </ref1> . . .
<usage2> INFO </usage2>
<meaning2> INFO </meaning2> . . . .
</word>

Thus the lexicon that was readable only by an intelligent scholar, is made very simple in structure in order to be understood by a novice in Sanskrit. This structure enabled us to develop a searchable e-dictionary with different kinds of searchable options.

8. COMPLEXITIES IN TAGGING

Since the source text was very much unstructured from computational aspects though it was humanly understandable, it was not so easy to tag the information in order to get a tree structure. There was no standard and common sequence of information for all entries. The following are some examples of the complexity of the text.

<word>
<cat>INF</cat>
<m1>,<m2>. . . <mn>
<ref1><ref2>. . . .<refn>
<ex1><ex2>. . . ..<exn>
</word>

However it sometimes goes as follows
<word>
<cat>INF</cat> <gr>INF</gr>
<m1>INF<ref1>INF</ref1><m1>
<m2>INF<ref2>INF</ref2><gr>INF<gr><eg>INF</eg><m2
<m3>. . . . . . <mn>
</word>

Further in some places the text has following structure
<word>
<cat>INF</cat> <gr><m1>
<m2><gr><m3><m4><ref2><ref3>
"source"<m4><gr>. . .
</word>

Even in grammar information there is no common way for representing the same. For example
<word>
<gr>"root" – "meaning of root" "suffix"</gr>
INF
</word>
<word>
<gr>"root" – "meaning of root" "suffix" "meaning of suffix"</gr>
INF
</word>
<word>
<gr>"root" – "meaning of suffix" "suffix"</gr>
INF
</word>

9. E-VAACASPATYAM ON CD ROM

The first version of Vaacapspatyam CD ROM is ready for public release with 6 kinds of search facilities. All 42,000+ word entries are indexed alphabetically. By selecting any word in the word-index, one can access information related to the selected word. In the second option, the user can enter any string he wants to search, by clicking on soft key board designed for Sanskrit alphabets. If the string is available in key word list, Machine calls for relevant information about the string entered by the user. The third search option helps the user in searching all related words to a particular concept like wordnet. This option is unique as it helps the user in getting all the related words while composing poems etc. The Fourth search option is yet another unique experiment for Sanskrit Manuscript editors. In this option two entry boxes

are given, where the user can specify the starting and ending letters of the word missing in the manuscripts. The machine brings all the possible words that begin and end with the specified letters. This option is found very much useful for the editors, while reading damaged manuscript with missing letters and words.[1] Another search option is also given for the user to search for usages and expressions taken from various texts for a particular string or word. One can even search for all words derived from a root or a word with any particular suffix. In addition to all these, word-game enriches the CD with an added value.

We hope that users of this CD-ROM will find it useful for their research and other applications. We also hope that this becomes a model for tagging of any Sanskrit or other Indian language lexicons.

10. TECHNICAL INFORMATION

The CD-R presentation is developed using Visual Basic. The system tools that are available in VB are used. To avoid problems in font display, the textual output is shown in Netscape browser (Version 4.5) using DV-TT-yogesh font developed for iscii text. This method is a tested one and works in all platforms from windows-98 to windows-XP and gets rid of broken font display problem while presenting the text through the browser. At the development stage Perl was extensively used for tagging manually annotated text and for removing errors.

11. FURTHER SCOPE FOR RESEARCH

It is needless to establish that such a work of developing e-lexicon added with information meta-tags is of high importance in language processing. However, the traditional Sanskrit lexicons are rich in content and poor in organization from computational aspects. They need to be restructured for computational purposes. This work poses several challenges for lexicographic science in computational linguistics. There are dozens of such complicated dictionaries like. “Sabdakalpadruma and V.S.Apte dictionary for Sanskrit-English vise-versa. Our team has started working on these both dictionaries. We hope we will come with good results very soon.

12. ACKNOWLEDGEMENT

The authers are thankful to the Vice Chancellor, RSVidyapeetha,Tirupati, K.V. Ramakrishnamacharyulu, Amba P.Kulkarni, Deeptha, Anilkumar, Administrative Heads of Vidyapeetha and Referees of the paper.

13. REFERENCES

1. Taranatha Tarka Vachaspati, Vachaspatyam, (Reprint) Rashtriya Sanskrit Sansthan, New Dehli, 2000.

2. Descartes and Bunce, Programing the Perl DBI, O’Reilly 2000.

3. Erik T. Ray & Jason McIntosh, XML and perl, O’reilly 2002.

4. Jeffrey E.F. Friedl ,Mastering Regular Expressions, O’Reilly 2002.

14. APPENDIX



Figure 1: Home page of Vacaspatyam

[1] See Appendix – II for Search option images in CD-ROM.



Figure 2: Alphabetical-index

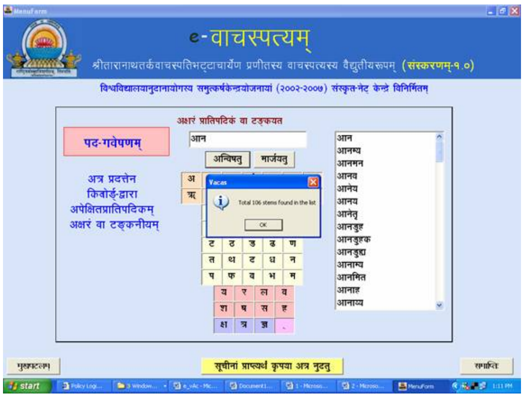


Figure 3: Word-Index



Figure 4: Editor's help

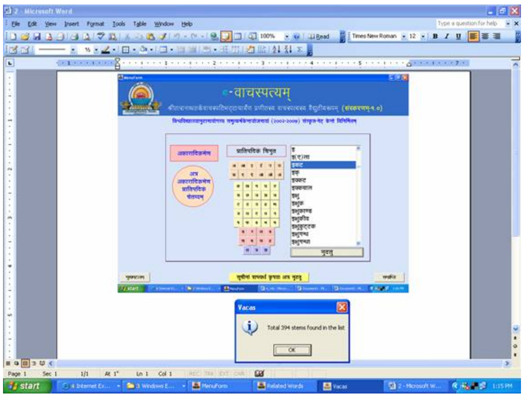


Figure 5: Grammatical Specialities-1



Figure 6: Grammatical Specialities-2



Figure 7: Grammatical Specialities-3

Critical Edition of Sanskrit Texts

Marc Csernel
INRIA, Projet AXIS,
Domaine de Voluceau,
Rocquencourt BP 105
78153 Le Chesnay Cedex, France
Marc.Csernel@inria.fr

François Patte
UFR de Mathématiques et Informatique,
Université Paris Descartes,
45 rue des Saints-Pères,
75270 Paris cedex 06, France
Francois.Patte@math-info.univ-paris5.fr.

ABSTRACT

A critical edition takes into account all the different known versions of the same text in order to show the differences between any two distinct versions. The construction of a critical edition is a long and, sometimes, tedious work. Some software that help the philologist in such a task have been available for a long time for the European languages. However, such software does not exist yet for the Sanskrit language because of its complex graphical characteristics that imply computationally expensive solutions to problems occurring in text comparisons.

This paper describes the Sanskrit characteristics that make text comparisons different from other languages, presents computationally feasible solutions for the elaboration of the computer assisted critical edition of Sanskrit texts, and provides, as a byproduct, a distance between two versions of the edited text. Such a distance can then be used to produce different kinds of classifications between the texts.

1. INTRODUCTION

A critical edition is an edition that takes into account all the different known versions of the same text. If the text is mainly known through a great number of manuscripts that include non trivial differences, the critical edition often looks rather daunting for readers unfamiliar with the subject.

- If the number of texts to compare is small and differences between texts are not too great, the text looks just like any commentated editions.
- If the text is mainly known through a great number of manuscripts that include non trivial differences, the critical edition looks often rather daunting for readers unfamiliar with the subject. The edition is then formed mainly by footnotes that enlighten the differences between manuscripts, while the main text (that of the edition) is rather short, sometimes a few lines on a page.

Note that in either case, the main text is established by the editor through his own knowledge. More explicitly, the main text can be either a particular manuscript, or a “main” text, built according to some specific criteria chosen by the editor.

Building a critical edition by comparing texts two by two, especially manuscript ones, is a task which is certainly long and, sometimes, tedious. This is why, for a long time, computer programs have been helping philologists in their work (see O’Hara (1993) or Monroy & *al.* (2002) for example), but most of them are dedicated to texts written in Latin (sometimes Greek) scripts. For example, the Institute for New Testament Textual Research (2006), at Münster University, provides an interactive critical edition of the Gospels.

In this paper we focus on the critical edition of manuscripts written in Sanskrit.

Our approach will be based on and illustrated by paragraphs and sentences that are extracted from a collection of manuscripts of the “Banaras gloss”, *kāśikāvṛtti* in Sanskrit (Kāśi is the name of Banaras). The Banaras gloss was

This paper was supported by the ACI CNRS “histoire des savoirs” and the Asia IT & C contract 2004/091-775.
Critical editions of the Gospels have induced a considerable amount of studies.

written around the 7th century A.D., and is the most widespread, the most famous, and one of the most pedagogical commentary on the notorious Pāṇini grammar.

Pāṇini's grammar is known as the first **generative** grammar and was written around the fifth century B.C. as a set of rules. These rules cannot be understood without the explanation provided by a commentary such as the *kāśikāvṛtti*. Notice that, since some manuscripts have been damaged by mildew, insects, rodents…, they are not all complete. In particular, they do not include all chapters; generally around fifty different texts are available for comparison at the same time.

In what follows we will first describe the characteristics of Sanskrit that matter for text comparison algorithms as well as for their classification. We will also present briefly the textual features we use to identify and to quantify the differences between manuscripts of the same Sanskrit text. We will show that such a comparison requires to use a lemmatized text as the main text.

Roughly speaking, lemmatization is a morpho-linguistic process which makes each word appear in its base form, generally followed by a suffix indicating its inflected form. For example *walking*, consists of the base form *walk*, followed by the suffix *ing* which indicates the continuous form. After a lemmatization each word will, at least, appear as separated from the others.

The revealed differences, which as a whole, form one of the most important parts of the critical edition, provide all the information required to build distances between the manuscripts. Consequently we will build phylogenetic trees assessing filiations between them, or any kind of classification regrouping the manuscripts into meaningful clusters. Finally, we will discuss the definition of a method of computation of faithful distances between any two Sanskrit texts, provided one of them is lemmatized.

2. HOW TO COMPARE SANSKRIT MANUSCRIPTS

2.1. Sanskrit and its graphical characteristics

One of the main characteristic of Sanskrit is that it is not linked to a specific script. A long time ago Sanskrit was mostly written with the Brāhmī script, but nowadays Devanāgarī is the most common one. Other scripts may be used, such as Bengali, in northern India, or Telugu, in southern India. In Europe, an equivalent (but fictive) situation would be to use either the Latin, Cyrillic, or Greek alphabets to write Latin. Sanskrit is written mostly with the Devanāgarī script that has a 48 letter alphabet.

Due to the long English presence in India, a tradition of writing Sanskrit with the Latin alphabet (a transliteration) has been established for a long time by many European scholars such as Franz Bopp (1816). The modern IAST — International Alphabet of Sanskrit Transliteration — follows the work of Monier-Williams in his 1899 dictionary. All these transliteration schemes were originally carried out to be used with traditional printing. It was adapted for computers by Frans Velthuis (1991), more specifically to be used with TEX. According to the Velthuis transliteration scheme, each Sanskrit letter is written using one, two or three Latin characters; notice that according to most transliteration schemes, upper case and lower case Roman characters have a very different meaning. In this paper, unless otherwise specified, a letter is a Sankrit letter represented, according to the Velthuis scheme, by one, two or three Latin characters.

In ancient manuscripts, Sanskrit is written without spaces, and from our point of view, this is an important graphical specificity, because it increases greatly the complexity of text comparison algorithms. One may remark that Sanskrit is not the only language where spaces are missing in the text: Roman epigraphy and European Middle Age manuscripts are also good examples of that.

2.2. The different comparison methods

Comparing manuscripts, whatever the language, can be achieved in two ways:

- When building a critical edition, the notion of word is central, and an absolute precision is required. For example, the critical edition must indicate that the word *gurave* is replaced by the word *gaṇeśāya* in some

manuscripts, and that the word *śrī* is omitted in others.

- When establishing some filiation relations between the manuscripts, or for a classification purpose, the notion of word can be either ignored, or taken into account. The only required information is the one needed to build a distance between texts. Texts can be considered either as letter sequences, or as word sequences.

Considering each text as a letter sequence, Le Pouliquen (2007) proposed an approach that determines the so called *"Stemma codicum"* (nowadays *filiation trees*) of a set of Sanskrit manuscripts. The first step consists in the construction of a distance according to the Gale and Church (1993) algorithm. This algorithm was first developed to provide sentence alignments in a multi-lingual corpus, for example a text in German and its English translation. It uses a statistical method based on sentence length. Gale and Church showed that the correlation between two sentence lengths follows a normal distribution. Once the distance is computed, a phylogenetic tree is built using the N-J —Neighbour-Joining— algorithm (Saitou and Nei (1987)).

On the other hand, each critical edition deals with the notion of word. Since electronic Sanskrit lexicons such as the one built by Huet (2004, 2006) do not cope with grammatical texts, we must find a way to identify each Sanskrit word within a character string, without the help of either a lexicon or of spaces to separate the words.

2.3. How shall we proceed?

The solution comes from the lemmatization of one of the two texts: the text of the edition. The lemmatized text is prepared **by hand** by the editor. We call it a *padapāṭha*, according to a mode of recitation where syllables are separated.

From this lemmatized text, we will build the text of the edition, that we call a *saṃhitapāṭha*, according to a mode of recitation where the text is said continuously. The transformation of the *padapāṭha* into the *saṃhitapāṭha* is not straightforward because of the existence of *sandhi* rules.

What is called *sandhi* — from the Sanskrit: liaison — is a set of phonetic rules which apply to the morpheme junctions inside a word or to the junction of words in a sentence. Though these rules are perfectly codified in Pāṇini's grammar, they could become quite tricky from a computer point of view. For instance, the final syllable *as* is mostly changed into *o* if the next word begins with a voiced letter, but the word *tapas* (penance) becomes *tapo* when it is followed by the word *dhana* (wealth) to build the compound *tapodhana* (one who is rich by his penances), while it remains *tapas* when composed with the suffix *vin*: *tapasvin* (an ascetic). What is a rule for Pāṇini, becomes an exception for computer programs and we have to take this fact into account.

A text with separators (such as spaces) between words, can look rather different (the letter string can change greatly) from a text where no separator are found.

We call the typed the text, corresponding to each manuscript, a *mātṛkāpāṭha*. Each *mātṛkāpāṭha* contains the text of a manuscript and some annotation commands.

`\gap`	Gap left intentionally by a scribe	`\deleted`	Text deleted by the scribe
`\afterc`	The text after a scribe's correction.	`\beforec`	The text before a scribe's correction
`\scribeadd`	Insertion made by the scribe without the presence of gap	`\eyeskip`	The scribe copying the text has skipped his eyes from one word to the same word later in the text.
`\doubt`	Text is not easily readable	`\inferred`	Text very difficult to read
`\lacuna`	The text is damaged and not readable	`\illegible`	Mainly concerns the deleted text

\insertioningap	Insertion made by a scribe in a gap	\foliochange	
\ignoredtext	This text of the manuscript, is not part of the opus	\marginote	Insertion made by the scribe, as his own commentary (but not part of the text)
\notes	Notes made by the scholar in charge of the collation		

Table 1: The collation commands.

These commands allow some information from the manuscript to be taken into account, but this information is not part of the text, such as ink colour, destruction, etc. They provide a kind of meta-information.

The typing of each *mātṛkāpāṭha* is done by scholars working in pair, one reading, one typing (alternatively). To avoid the typing of a complete text, they copy and modify the text of the *saṃhitapāṭha* according to the manuscript.

- **First step**: A twofold lexical preprocessing. First the *padapāṭha* is transformed into a virtual *saṃhitapāṭha* in order to make a comparison with a *mātṛkāpāṭha* feasible.

  The transformation consists in removing all the separations between the words and then in applying the *sandhi*. This virtual *saṃhitapāṭha* will form the text of the edition, and will be compared to the *mātṛkāpāṭha*. As a sub product of this lexical treatment, the places where the separation between words occurs will be kept into a table which will be used in further treatments (see: 4.4).

  On the other hand, the *mātṛkāpāṭha* is also processed, the treatment consists mainly in keeping the collation commands out of the texts to be compared. The list of the commands can be found in Table 1 (p. 106) with some explanation when needed. Notice that for practical reasons, these commands cannot, for the time being, be nested. Out of all these commands just a few have an incidence on the texts to be compared.

- **Second step:** An alignment of a *mātṛkāpāṭha* and the virtual *saṃhitapāṭha* (an alignment is an explicit one to one correspondence of the letters of the two texts.) A more precise definition can be found on page 111. The Longest Common Subsequence algorithm is applied to these two texts. The aim is to identify, as precisely as possible, the words in the *mātṛkāpāṭha*, using the *padapāṭha* as a pattern. Once the words of the *mātṛkāpāṭha* have been determined, we can see those which have been added, modified or suppressed.

The comparison is done paragraph by paragraph, the different paragraphs being constructed in each *mātṛkāpāṭha* by the scholar who collated them, according to the paragraph made in the *padapāṭha* during its elaboration. In a first stage, the comparison is performed on the basis of a Longest Common Subsequence. Each of the obtained alignments, together with the lemmatized text (i.e. the *padapāṭha*), suggests an identification of the words of the *mātṛkāpāṭha*. However, due to the specificities of Sanskrit, the answer is not straightforward, and a consistent amount of the original part of this work concerns this identification process. Surprisingly the different rules used for this determination are not based on any Sanskrit knowledge, but on common sense. The result of the application of these rules has been validated by Sanskrit philologists.

We remark that the kind of results expected for the construction of a critical edition (what words have been added, suppressed or replaced in the manuscript) is similar to the formulation of an edit distance, but in terms of *words*. The results we obtain from the construction of the critical edition can be transformed into a distance between the manuscripts.

3. THE LEXICAL PREPROCESSING

The goal of this step is to transform both the *padapāṭha* and the *mātṛkāpāṭha* in order to make them comparable. This treatment will mainly consist in transforming the *padapāṭha* into a *saṃhitapāṭha*. The *mātṛikāpāṭha* will be

purged of all collation commands, except some of the commands which modify the text to be compared, namely `\scribeadd`, `\afterc`, `\inferred`. All lexical treatments are build using `Flex`, a `Linux` version of `Lex` which is a free and widely known software.

At the end of the lexical treatment the text corresponding respectively to the *padapāṭha* and the *mātṛkāpāṭha* is transmitted to the comparison module with an internal encoding (see Table 4, p. 109). This allows us to ensure the comparison whatever the text encoding — unicode instead of Velthuis code for instance — the only condition is to build a new lexical scheme, which is a perfectly delimited work albeit a bit time-consuming.
An example of *padapāṭha*:

```
iti+anena krame.na var.naan+upa^di"sya+ante .na_kaaram+itam+|
```

we can see that words are separated by spaces and three different lemmatization signs: +, _, ^ which have the following meanings:

- **+**: Indicates a separation between inflected items in a sentence.
- _: Indicates a separation between non inflected items of a compound word.
- ^: Indicates the presence of a prefix; this sign is not, for the moment, taken into account for the comparison process. It will be used for a future automatic index construction.

3.1. The lexical preprocessing of the *mātṛkāpāṭha*

The main goal of this step is to remove the collation commands in order to keep only the text of the manuscript for a comparison with the *saṃhitapāṭha*. The list of these commands can be found in Table 1 (p. 106). The tables described hereafter follow more or less the `Lex` syntax, with a major exception, for readability reason: the suppression of the protection character denoted "\". We will note briefly some of the main features:

The character "|" means **or**; a name included within braces, such as `{VOWEL}`, is the name of a letter subset defined in Table 2 (p. 107). It can be replaced by any letters of the subset. The character "/" means **followed by**, but the following element will not be considered as part of the expression: it will stay within the elements to be further examined; examples of the use of the character "/" will be found hereafter in Table 3 (p. 108).

Note that some possible typographical errors induced us to remove all the spaces from the *mātṛkāpāṭha* before the comparison process. Thus no words of the *mātṛkāpāṭha* can appear separately during that process.

<table>
<tr><td>SOUR</td><td>k|kh|c|ch|.t|.th|t|th|p|ph|"s|.s|s|.h</td></tr>
<tr><td>NAS</td><td>n|.n|"n|~n|m|.m</td></tr>
<tr><td>VOWEL_A</td><td>aa|i|ii|u|uu|.r|.R|.l|.L|e|ai|o|au</td></tr>
<tr><td>VOWEL</td><td>a|{VOWEL_A}</td></tr>
<tr><td>DIPH</td><td>e|ai|o|au</td></tr>
<tr><td>CONS</td><td>k|kh|g|gh|"n|c|ch|j|jh|~n|.t|.th|.d|.dh|
.n|t|th|d|dh|n|p|ph|b|bh|m|"s|.s|s</td></tr>
<tr><td>SON</td><td>g|gh|j|jh|.d|.dh|d|dh|b|bh|l|r|y|v|{NAS}|.m|h</td></tr>
<tr><td>GUTT</td><td>k|kh|g|gh|"n</td></tr>
<tr><td>PALA</td><td>c|ch|j|jh|~n</td></tr>
<tr><td>LEMM</td><td>+|_|^</td></tr>
<tr><td>DENTA</td><td>t|th|d|dhn</td></tr>
<tr><td>LABIA</td><td>p|ph|b|bh|m</td></tr>
</table>

Table 2: Some lexical definition of letter categories.

Table 2 provides a definition for the subset definition such as `VOWEL_A` defined in the third line, which

is a subset of the alphabet containing all the vowels except **a**, in fact one of the following letters:
aa, i, ii, u, uu, .r, .R, .l, .L, e, ai, o, au
according to the Velthuis encoding scheme, and `VOWEL`, next line, defined by any letter in: `a|{VOWEL_A}` can be any letter in `VOWEL_A` or `a`. Notice that the subset `LEMM` contains the different lemmatization signs found in the *padapāṭha*.

Table 3 (p. 108) contains some example of **generative** *sandhi* where a new letter (or a sequence of letters) is inserted within the text. Table 5 (p. 109) contains some examples of ordinary *sandhi* where a set of letters is replaced by one or two other letters.

The contents of both preceding tables will be explained in the following section.

3.2. The lexical preprocessing of the *padapāṭha*

The main goal of this step is to apply the *sandhi* rules in order to transform the *padapāṭha* into a *saṃhitapāṭha*, the other goal is to purge the *padapāṭha* of all unwanted characters. The *sandhi* (p. 105) are perfectly determined by the grammar of Sanskrit (see for example (Renou (1996)). They induce a special kind of difficulties due to the fact that their construction can be, in certain cases, a two-step process. During the first step, a *sandhi* induces the introduction of a new letter (or a letter sequence). This new letter can induce, in the second step, the construction of another *sandhi*. The details of the lexical transformation expressed as a `Flex` expression can be found in Table 3 (p. 108) for the first step, and in Table 5 (p. 109) for the second one.

`as+/{SON}`	`Add("o"); AddSpace();`
`as+/{VOWEL_A}`	`Add("a"); AddSpace();`
`as+a`	`Add("o.a");`
`aas+/{VOWEL}`	`Add("aa"); AddSpace();`
`as+/(k|p|s|.s|"s)`	`Add("a.h"); AddSpace();`
`ai/+{VOWEL}`	`Add("aa"); AddSpace();`
`ai(_|^)/{VOWEL}`	`Add("aay");`

Table 3: Some of the generative *sandhi*.

Table 3 can be read in the following way: the left part of the table contains a `Flex` expression, the right part some procedure calls. The two procedures are `Add("xxx")`, which adds the letter sequence `xxx` to the text of the *padapāṭha*, and `AddSpace()` which adds a space within the text. When the expression described in the left part is found within the *padapāṭha*, the procedures described in the right part are executed. The letters belonging to the expression on the left of the sign "/" are removed from the text. The different expressions of the left part are checked according to their appearence. The tests are done in sequential arrangement.

For example the first three lines of Table 3 state that:

- If a sequence `as`, followed by a lemmatization sign `+`, is followed by any letter of the `{SON}` subset defined in Table 2, the program puts in the text an `o` followed by a space; the letter sequence `as+` will be dropped out from the text, but not the element of `{SON}`.

  Example: If the sequence `bahavas+raa"sayas+hataas+|` is found in the *padapāṭha*, the sequence `as+/{SON}`: `bahavas+r` and `raa"sayas+h` is found twice.
  Therefore, according to the rules defined in the right column of the table, we get as a result in the *saṃhitapāṭha*: `bahavo_raa"sayo_hataa.h|`, corresponding to the Sanskrit text: *bahavo rāśayo hatāḥ* |
- If the sequence `as+` is followed by a letter which belongs to `{VOWEL_A}`, an `a` will be generated and the element belonging to `{VOWEL_A}` will remain.

The case `hataas+|` is not one of these for two reasons: 1) `aas+` is different from `as+`, according to the Velthuis encoding scheme, 2) `|` is a punctuation mark and does not belong to the category `{SON}`, it has its own way of treatment.

Example: If the sequence `prakalpitas+i.s.taraa"sis+|` is found in the *padapāṭha*, we have one sequence `as+/{VOWEL_A}`: `prakalpitas+i` and, in this case, the program will return within the *saṃhitapāṭha*: `prakalpita_i.s.taraa"si.h|`. The Sanskrit text: *prakalpita iṣṭarāśiḥ* |

- If the sequence `as` is followed by a lemmatization sign + and by the letter `a`, it will be replaced by the sequence `o.a` and no space will be added.

  Example: If the sequence `yogas+antare.nonayutas+ardhitas+|` is found in the *padapāṭha*, the sequence appears twice: `yogas+a` and `yutas+a`; this will be changed into: `yogo.antare.nonayuto.ardhita.h|`, corresponding to the Sanskrit text: *yogo'ntareṇona-yuto'rdhitaḥ*|

Once Table 3 has been used with the *padapāṭha*, Table 5 and Table 4 are used in the same lexical pass.

Table 4 is really simple: the left part contains a character sequence corresponding to the Velthuis code, the right part contains a `return` code followed by an upper case letter sequence beginning by an `L`. This letter sequence is the name of an internal code that corresponds to a *devanāgarī* letter and will be used for further treatment.

e	return LE;
ai	return LAI;
aa	return LABAR;
au	return LAU;
k	return LK;
"n	return LNQU;
~n	return LNTI;
.n	return LNPO;

Table 4: Examples of Velthuis characters encoding, with linked internal code

Table 5 is a little bit more complicated in its right part. It contains references to two variables `Alter` and `Next` and each of these variables is affected by a value of the internal code corresponding to the Velthuis code.

.m/{GUTT}	Alter = LNQU; return LMPO;
.m/{PALA}	Alter = LNTI; return LMPO;
.m/{DENTA}	Alter = LN; return LMPO;
.m/{LABIA}	Alter = LM; return LMPO;
(a|aa){LEMM}(a|aa)	return LABAR;
(a|aa){LEMM}(o|au)	return LAU;
.r/{LEMM}({VOWEL}|{DIPH})	return LR;
e{LEMM}a	Next = LAVA; return LE;
o{LEMM}a	Next = LAVA; return LO;
(k|g)/{LEMM}{SOUR}	return LK;
(k|g|c)/{LEMM}({SON1}|{VOWEL})	return LG;
(k|g|c)/{LEMM}{NAS}	Alter = LNPO; return LG;
(.t|.d|.s)/{LEMM}{SOUR}	return LTPO;
(.t|.d|.s)/{LEMM}{NAS}	Alter = LNPO; return LDPO;
(.t|.d|.s)/{LEMM}({SON}|{VOWEL})	return LDPO;
as/+" "	Next = LHPO; return LA;

Table 5: Some normal *sandhi*

The variable Alter corresponds to an alternate value to the returned code (in other terms, the code of another possible letter), the variable Next corresponds to the code letter generated by the *sandhi* which will **always** follow the returned letter. If Alter take a value, the letter is equivalent to the letter returned by the normal process so, the returned and the Alter value can be exchanged and the distance between the letters is zero.

The first four lines treat the letter .m — *ṃ, anusvāra* — in different contexts: if this letter is followed by a letter belonging to one of the subsets GUTT, PALA, DENTA, LABIA, defined in Table 2, there could be, in some manuscript, an alternate letter for it. This is mainly due to scribe habits and we must make the software aware of this. For instance the word *aṅka* can also be written *aṃka*, in which case we are in the situation .m/GUTT; according to the instruction: Alter=LNQU; return LMPO;, while comparing our virtual *saṃhitapāṭha* with a *mātṛkāpāṭha*, if, at the same place in two *mātṛkāpāṭha*, the comparison software reads a.mka or a"nka, no variant will be reported and the value of the distance distance between a.mka or a"nka is zero. A similar situation occurs for the readings pa.n.dita/pa.m.dita (.m/PALA) or sandhi/sa.mdhi (.m/DENTA) or sambhuu/sa.mbhuu (.m/LABIA).

The variable Next is used whenever the *sandhi* rule induces the production of a new character next to the character (or string) concerned by the *sandhi*.

For instance, if we have: tanmuule+a.s.tayute, the e before the lemmatization sign will remain, but the a will be elided and replaced by an *avagraha*; this is the meaning of the rule in line 8: if we have e{LEMM}a then e is kept: return LE and next to it an *avagraha* is produced: Next=LAVA; so we get: tanmuule.a.s.tayute.

The same procedure is done with the last line of the table: if a word is ended by as and followed by a blank space, as is dropped (meaning of "/"), a is returned followed by a *visarga*: Next=LHPO.

Line 6 contains the premises of further difficulties: it states that the letter a or aa followed by a lemmatization sign and the by the letter a or the letter aa (correponding to the sanskrit letter *a* and *ā* in traditional transliteration) will become the LABAR code (corresponding to the letter aa: *ā*). Two letters and the lemmatization sign will become a single letter. Consequently, if a variant occurs which concerns the letter aa the program will not know if the variant concerns the word of the *padapāṭha* before or after the lemmatization sign.

First example. If we have in the *padapāṭha*: "sabda_artha.h, it will become "sabdaartha.h in the *saṃhitapāṭha*. If we have in the *mātṛkāpāṭha*: sabde.artha.h, the program will have to decide between some of the following possible solutions:

"sabda has been changed into "sabde and artha.h has been changed in .artha.h

"sabda has been changed into "sabd and artha.h has been changed in e.artha.h

"sabda has been changed into "sabde.a and artha.h has been changed in rtha.h

Second example. If we have the *padapāṭha*: asya+artha.h, it will become asyaartha.h in the *saṃhitapāṭha*. If we have in the *mātṛkāpāṭha*: asyaa artha.h, as the program, in the lexical preprocessing, removes the spaces in the *mātṛkāpāṭha*, it will have to decide between some of the following possible solutions:

asya has been changed into asyaa and artha.h stays unchanged.

asya has been changed into asyaa and artha.h has been changed in rtha.h.

Third example. If we have the *padapāṭha*: iti+u_kaare.na+a_kaara_aadaya.h, it will come in the *saṃhitapāṭha* as ityukaare.naakaaraadaya.h. If we have in the *mātṛkāpāṭha*:

ityukaare.nekaaraadaya.h, the comparison can be very confusing because one word is completely missing: the a in a_kaara. This creates a very important problem: we had not imagined at the beginning of our work that a complete word could disappear if only one letter was missing.

4. COMPARING THE SANSKRIT MANUSCRIPTS WITH THE TEXT OF THE EDITION

In this section we will come to the heart of our research. We compare, sentence by sentence, the text of each *mātṛkāpāṭha* (i.e. a collated manuscript), purged of every collation commands, with the *padapāṭha* transformed into a *saṃhitapāṭha*.

For each comparison we start to **align** each *mātṛkāpāṭha* sentence, word by word, with those of the *saṃhitapāṭha*. This comparison uses the word limits provided by the lemmatization done in the *padapāṭha*. We use a basic tool: the Longest Common Subsequence (LCS) algorithm to begin our alignment process.

In the following, we describe the LCS algorithm by giving an example. Then we explain why the use of the LCS still raises some problems. We can solve some on these problems by carefully sailing through the LCS matrix, thanks to the limits provided by the *padapāṭha*.

Even with such a help, and a careful navigation through the solution spaces, we have to keep track of a various number of possible solutions in order to compute a score attached to each possible solution. This score allows us to choose the most suitable one.

Roughly speaking, an alignment between two characters string `A` and `B` is a one to one correspondence of the characters of `A` with the characters of `B` or with the empty character denoted "_". The alignment process is symmetrical. Generally different possible alignments exist between two strings. Table 6 give three different possible alignments between A = **aaabbb** and B = **aaacbb**.

a	a	a	_	b	b	b
a	a	a	c	_	b	b

a	a	a	_	b	b	b
a	a	a	c	b	_	b

a	a	a	b	b	_	b
a	a	a	_	c	b	b

Table 6: Examples of possible alignments

4.1. The Longest Common Subsequence algorithm.

The Longest Common Subsequence (LCS) algorithm is a well-known algorithm used in string sequence comparison. The goal of this algorithm is to provide a longest common substring between two character strings.

More precisely, given a sequence $X = \langle x_1, x_2, ..., x_m \rangle$, another sequence $Z = \langle z_1, z_2, ..., z_n \rangle$ is a subsequence of X if there is a strictly increasing sequence of indices $\langle i_1, i_2, ... i_k \rangle$ such that $z_j = x_{i_j}$ for each $j \in [1 : k]$. For example, if $X = \langle A, B, C, D, A, B, C \rangle$ then $Z = \langle B, D, B, C \rangle$ is a subsequence of X. A common subsequence to sequences X and Y is a subsequence of both X and Y. Generally there is more than one LCS. We denote $|X|$ the length of X, and $X[i]$ the i^{th} character of that sequence.

Computing the LCS is equivalent to computing an edit distance between two character strings. An edit distance between sequences X and Y is the minimum number of operations such as suppression, addition and replacement (in term of characters) needed to change the sequence X into Y. An edit distance that is computed without the replacement operation is sometimes called *LCS distance* by some authors. This function is a kind of dual length of the length of an LCS between X and Y(see, for more details, Crochemore *et al.* (2001), chapter 7). The length of a LCS between X and Y will be denoted $lcs(X, Y)$ or simply lcs if there is no ambiguity. The edit distance and the LCS can be computed efficiently by the dynamic programming algorithm.

Once the computation of an lcs is achieved, one can compute an alignment of the two sequences. Most of the time, one considers any of the alignments as equivalent. It will not be the case here, because the comparison is based on words, not only on characters.

Example 1. Let us compute the *lcs* between two (simple) Sanskrit texts: $X =$ `yamaan`, $Y =$ `yamin`. Note that according to the Velthuis transliteration `aa` is a single letter: long `a` (ā).

		y	a	m	i	m
	0	0	0	0	0	0
y	0	1	1	1	1	1
a	0	1	2	2	2	2
m	0	1	2	3	3	3

The Unix `diff` command is based on this algorithm.

aa	0	1	2	3	3	3
m	0	1	2	3	3	4

Table 7: Computation of an LCS matrix T.

The value of the lcs, here 4, is displayed at the bottom right corner of the matrix T. The distance between the two sequences is $d(X,Y) = |X| + |Y| - 2 * lcs(X,Y)$. In this exemple $d(X,Y) = 5 + 5 - 2 * 4 = 2$ (the letter `m` is suppressed and the letter `aa` is added).

The matrix is initialised to zero, and each score is computed by:

$$T[i,j] = \begin{cases} T[i-1,j-1]+1 & \textbf{if } X[i] = Y[j], \\ max\{T[i-1,j], T[i,j-1]\} & \textbf{otherwise.} \end{cases}$$

The score $T[i,j]$ gives the value of the lcs between subsequences $X[1:i]$ (the i first characters of the sequence X) and $Y[1:j]$. These subsequences are defined as the first i letters of X and j letters of Y respectively. Each score $T[i,j]$ can be computed using some adjacent scores as shown in the previous formula. The complexity of the matrix computation is obviously in $O(|X||Y|)$. In this example, the LCS matrix generates exactly the two following symmetrical alignments.

y	a	m	i	_	m
y	a	m	_	aa	m

y	a	m	_	i	m
y	a	m	aa	_	m

Table 8: The two possible alignments.

The alignment can be read in the following way: when letters are present in the same column of the two rows, they belong to the LCS. When a letter `l` is present with an opposite “-”, then `l` can be considered either as added in the line where it appears, or suppressed from the line where the opposite “-” is present.

Example 2. The comparison between two short sentences, as shown in Figure 1, describes the way we proceed and what kind of result can be expected. The sentences compared in this example are:
tasmai śrīgurave namas and *śrīgaṇeśāya namaḥ*, which are encoded:

`tasmai "srii_gurave namas` and `"sriiga.ne"saaya nama.h`

Note that the first sentence (X) belongs to the *padapāṭha*, the second (Y) to a *mātṛkāpāṭha*, and that the character “_” (underscore) is a lemmatization sign.

		"		i			.		"	a							.
		s	r	i	g	a	n	e	s	a	y	a	n	a	m	a	h
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
s	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
m	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2
ai	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2
"s	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
r	0	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ii	0	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
g	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4
u	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4
r	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4
a	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	5
v	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	5
e	0	1	2	3	4	5	5	6	6	6	6	6	6	6	6	6	6
n	0	1	2	3	4	5	5	6	6	6	6	6	7	7	7	7	7
a	0	1	2	3	4	5	5	6	6	6	6	7	7	8	8	8	8
m	0	1	2	3	4	5	5	6	6	6	6	7	7	8	9	9	9
a	0	1	2	3	4	5	5	6	6	6	6	7	7	8	9	10	10
.h	0	1	2	3	4	5	5	6	6	6	6	7	7	8	9	10	11

Figure 1: A second example.

The matrix in Figure 1 contains all the possible alignments, one of them being the alignment in Table 9. We can see that the string `tasmai` is missing in the *mātṛkāpāṭha*, that the string `"srii` is present in both sentences, that `gurave` is replaced by `ga.ne"saaya`, and that the string `nama.h` is present in both sentences but under two different aspects: "`nama.h`" and "`namas`". The rule that states the equivalence between character `.h` and character `s` is one of the *sandhi*'s (see: 3.2). The following alignment is one of the possible results, the separation between words of the *padapāṭha* being represented by double vertical lines.

We can see in this example that the value of the *lcs*(X, Y) is 14 and it appears in the right bottom corner of the table. The distance between X and Y expressed in terms of letters is:

$$d(X, Y) = |X| + |Y| - 2 * lcs(X, Y) = 16 + 19 - 2 * 14 = 7$$

In terms of words, one word is missing: `tasmai`; the word `gurave` can be considered as replaced by `ga.ne"saaya` or missing in the *padapāṭha* and `ga.ne"saaya` added in the *mātṛkāpāṭha*. The value of the distance in terms of words will be either two or three according to the definition of the replacement operation.

t	a	s	m	ai	"s	r	ii	g	u	r	a	-	v	e	-	-	-	-	n	a	m	a	s
-	-	-	-	-	"s	r	ii	g	-	-	a	.n	-	e	"s	aa	y	a	n	a	m	a	.h

Table 9: The corresponding alignment.

During our comparison process, we must keep in mind that our final goal is to provide a difference between a *mātṛkāpāṭha* and the *padapāṭha* in terms of words. To appreciate the quality of this difference, an implicit criterion is to say that **the fewer words concerned, the better the criterion**, all things being equal, the word boundaries being provided by the *padapāṭha*.

Consequently, in what follows we will choose, whenever possible, the solution which not only minimises the number of words concerned, but also, as far as no other criteria are involved, minimises the number of letters concerned.

```
1c1
< "sriigane"saayanama.h
---
> tasmai"sriiguravenama.h
```

`diff` without space

```
1d0
< tasmai
4c3,5
< gurave
---
> gane
> "
> saaya
```

`diff` with space

```
Word 1 'tasmai' is :
- Missing
Word 2 '"srii' is :
- Followed by Added word(s)
'ga.ne"saaya'
Word 3 'gurave' is :
- Missing
```

Our results without space

Table 10: different comparisons

4.2. Why not use the `diff` algorithm

The authors very first idea was to use `diff` in order to obtain the differences between two sanskrit sequences. It is stated in `diff` documentation that the inspiration of the actual version of `diff` was provided by the paper of Myers (Myers 1986).

But the results were quite disapointing. The classical `diff` command line provided no useful information at all. The result of the comparison of the two following sequences: `"srii ga.ne"saaya nama.h` and `tasmai "srii_gurave namas` just said that they were different.

We obtained a slightly better result with `Emacs ediff`, as shown in Table 10, middle column: we can see which words are different. But as soon as we wanted to compare the same sequences without blank, we could not get a better result using `ediff` than using `diff`. This is why we started to implement our own algorithm. Its results appear in the right column of Table 10. We can see that they are expressed in term of words.

- Concerning `diff` and Myers's paper and all the derivated litterature, the emphasis is lain on the performance, for time as well as for space.
- Concerning our algorithm, no optimization has been applied, the main goal is to use the *padapāṭha* as a template on a *mātṛkāpāṭha* to determine, as well as possible, the end of words. Once we have determined the one to one correspondance between the words of the *mātṛkāpāṭha* and of the *padapāṭha*, we are nearly finished and there only remains to compare two Sanskrit letter strings to see their differences. Obviously, the added or missing words have to be noted carefully.

4.3. Sailing through the LCS matrix

The LCS matrix is only a base for further computations. What we need is an alignment which can provide us with some reasonable results. Each alignment corresponds to a path within the matrix. A short explanation of the construction of an alignment can be found in the first chapter of (Charras & Lecroq (website)) or in (Crochemore 2003).

The matrix provides alignments coming from the rightmost lowest corner to the leftmost upper corner (inverse order from the usual reading direction) in the following way:

1. if $\boldsymbol{T}[i,j] < \boldsymbol{T}[i+1,j+1]$ and if $\boldsymbol{X}[i] = \boldsymbol{Y}[j]$ we move (left and up) from $\boldsymbol{T}[i+1,j+1]$ to $\boldsymbol{T}[i,j]$ and in this case, the score, which is decreased by 1, indicates that a (common) letter has been added to the left of the alignment. $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{X}[i] \\ \boldsymbol{Y}[j] \end{pmatrix} .\boldsymbol{A}$ (the dot indicates the concatenation operation).
2. otherwise, if $\boldsymbol{T}[i,j] < \boldsymbol{T}[i,j+1]$ we move vertically up one row and add $\begin{pmatrix} \boldsymbol{X} \\ - \end{pmatrix}$ at the beginning of the alignment A = $\begin{pmatrix} \boldsymbol{X} \\ - \end{pmatrix} .\boldsymbol{A}$.

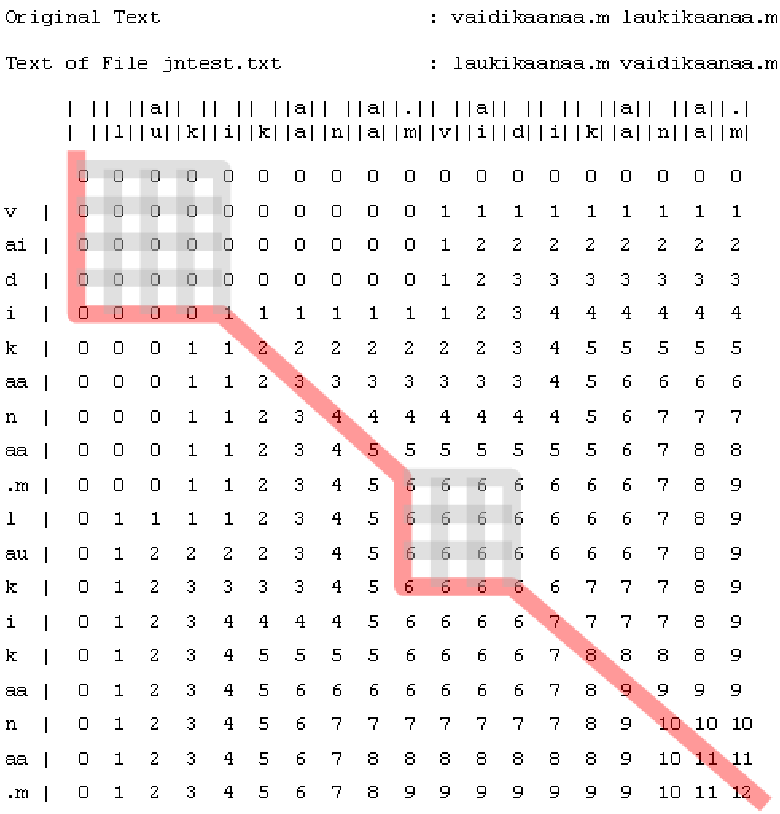


Figure 2: The different alignments within the matrix.

3. otherwise, we move horizontally one column left. In this case, add $\begin{pmatrix} - \\ X \end{pmatrix}$ to the alignment.

Figure 2 (p. 115) presents all the alignments provided by the LCS algorithm in an LCS matrix. The dark grey line depicts the chosen alignment, and the light grey lines represent other alignments also provided by the LCS algorithm. The sequence X belonging to the *padapāṭha*, the alignments are selected in order to maximise the number of consecutive letters belonging to X. This choice reduces the risk for two parts of the same word in the *padapāṭha* to be identified with two different subsequences of the *mātṛkāpāṭha*.

The chosen alignment corresponding to the dark grey line is depicted in Table 11.

v	ai	d	i	_	_	_	_	k	aa	n	aa	.m	l	au	k	_	_	_	i	k	aa	n	aa	.m
_	_	_	_	l	au	k	i	k	aa	n	aa	.m	_	_	_	v	ai	d	i	k	aa	n	aa	.m

Table 11: The chosen alignment.

It may be pointed out that when the different paths through the matrix form a square (no common letters can be found between X and Y at this place), the number of possible alignments grows very quickly. If N is the size of the square, the number of different alignments generated by each square is:

$$\binom{2N}{N} = \frac{(2N)!}{N!N!}$$

To provide a good idea of the possible number of paths, if we have a matrix which contains two ten by ten squares we get approximately 39×10^9 different possible alignments. This number expresses how complicated the comparison of Sanskrit texts is, and excludes any method that would require to examine all the possible alignments produced by the LCS algorithm.

4.4. Optimization: some navigation rules

In order to restrict the number of alignments, we will provide some navigation rules within the matrix. These navigation rules will greatly limit the number of solutions to be considered but they are unable to provide a good solution by themselves. Other steps are necessary to obtain a solution which gives some satisfaction to the philologists.

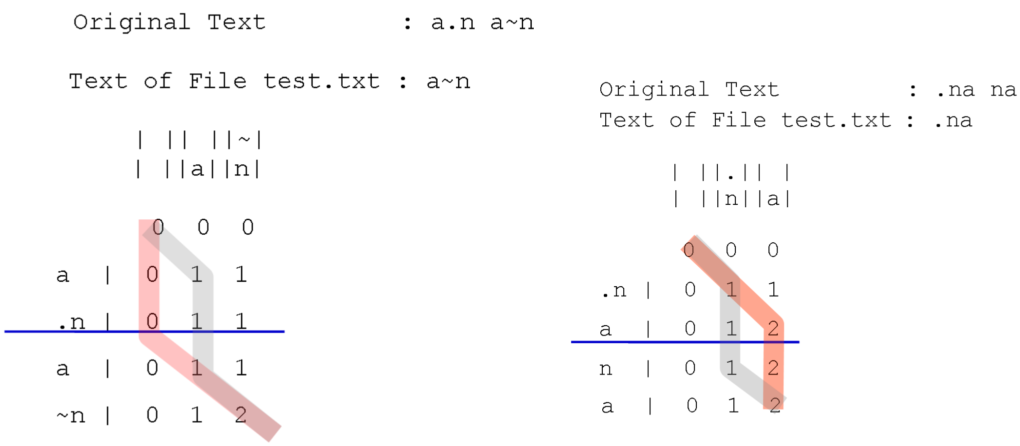
Let us try to give an idea of the different navigation rules implemented within the program. They concern the best way to choose a path (corresponding to an alignment) in the LCS matrix. Though in the preceding paragraph we described, for mathematical reason, the navigation through the matrix in the upward direction, right to left, we will now describe this navigation in the usual order, downward, and left right, which is easier to understand.

As a first remark we must notice that when the different paths form a square which corresponds to a place where there is no letter in common between the strings X and Y, we always go down first as in table 2. It induces to write in the alignment the part of the *padapāṭha* corresponding the square first, then write the corresponding part of the *mātṛkāpāṭha*.

But the great question we will always keep in mind during the navigation through the matrix is: **shall we align the soonest or the latest sequence**? The answer to this question will determine the navigation rules.

Table 12 shows two examples, the left one needs to be aligned **the latest** in order to provide the good result, on the contrary, the right one needs to be aligned **the soonest**. In each figure, the right path will be displayed in dark grey, the wrong one in light grey.

- On the left example, we see the comparison between two strings, in the *mātṛkāpāṭha*: `a˜n` and in the *padapāṭha*: `a.n a˜n`. The LCS matrix is displayed in Table 12 a) and the corresponding alignment in Table 14. The left solution in the table is the best according to common sense, it is also the best according to our criterion: **the fewer words concerned, the better the criterion**. The conclusion of this alignment is: the string `.n` is missing in the *mātṛkāpāṭha*.
- On the right example we see the comparison between two strings, in the *mātṛkāpāṭha*: `.na` and in the *padapāṭha*: `.na na`. The LCS Matrix is displayed in Table 12 b) and the corresponding alignment in Table 13, the left one is the best according to common sense, it is also the best according to our criterion. The conclusion of this alignment is: the string `na` is missing in the *mātṛkāpāṭha*.



a) Align the latest b) Align the soonest

Table 12

Our examples are sometimes taken from Sanskrit manuscripts, sometimes built for demonstration purpose, without any Sanskrit meaning.

a	.n	a	˜n
a	_	_	˜n

the light grey line (bad)

a	.n	a	˜n
_	_	a	˜n

the dark grey line (good)

Table 13: Align the latest

.n	a	n	a
.n	_	_	a

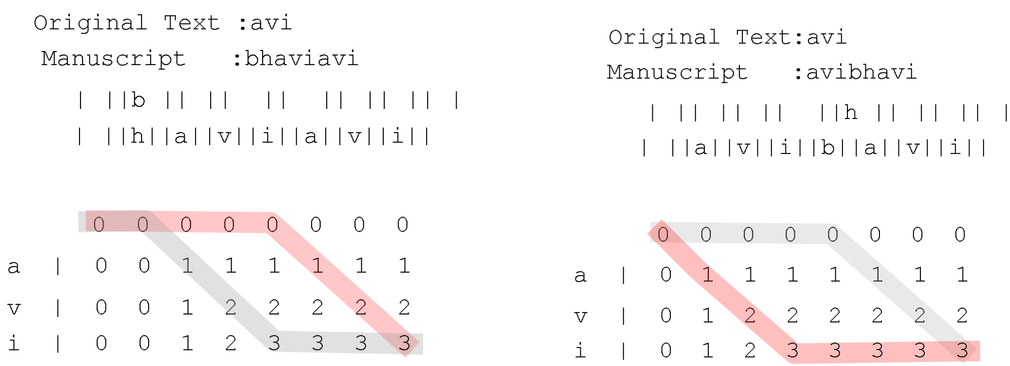
the light grey line (bad)

.n	a	n	a
.n	a	_	_

the dark grey line (good)

Table 14: Align the soonest

Our second example is different with a *mātṛkāpāṭha* involving more letters than the *padapāṭha*. The corresponding alignments can be seen in Table 16. Only the **good** alignments are displayed.



a) rule 2 b) rule 3

Table 15

_	_	_	_	a	v	i
bh	a	v	i	a	v	i

Align the latest.

a	v	i				
a	v	i	bh	a	v	i

Align the soonest.

Table 16:

What kind of conclusion can we draw from these apparently contradictory samples?

1. By default align the latest.
2. If, while aligning the soonest, we cross one of the *padapāṭha* word boundaries, then align the soonest.
3. If the choice occurs at the end of a *padapāṭha* word, then align the latest without further checking.
4. If, while aligning the soonest, we cross one of the *padapāṭha* word boundaries, then align the soonest.

The limit of words which are determined by the *padapāṭha* are the major determinant of the navigation rules. The rules displayed here are not complete, others exist (not described here), but they are more or less based on the same principles.

4.5. Improvement of the initial LCS alignment by the use of a score

As the first author of this paper has absolutely no knowledge of Sanskrit, he was looking for evaluating this result, and he found that our first criterion **if fewer words are concerned, the criterion is better** must be followed by another one: the **compactness** of the alignment.

The following example provides an idea of what we expect:

.r	k	aa	r	e	e	v	a	a	c	k	aa	r	y	aa	.n	i
.r	_	aa	r	_	_	_	_	_	_	_	_	_	y	aa	.n	i

The alignment has been built according to the navigation rules. It can be interpreted as: the word *kāre* is replaced by *ār*, the words *eva* and *ac* are missing, the word *kāryāṇi* is replaced by *yāṇi*.

.r	k	aa	r	e	e	v	a	a	c	k	aa	r	y	aa	.n	i
.r	_	_	_	_	_	_	_	_	_	_	aa	r	y	aa	.n	i

The second alignment is built taking compactness into account, it can be interpreted as: the words *kāre*, *eva* and *ac* are missing, the word *kāryāṇi* is replaced by *āryāṇi*, (the letter k is missing), which is obviously the best solution.

4.6. Problems which cannot be solved by the LCS

The identification of words in the *mātṛkāpāṭha*, as implicitly defined from the previous alignments, is not completely satisfactory. Indeed the maximisation of the lcs cannot fulfill our purpose, because the value of the lcs is only related to the notion of character, whereas our aim is to compare the texts word by word. Once the alignment is obtained, the words of the *mātṛkāpāṭha* are not really identified. To improve this alignment we propose a procedure which consists in local changes of the alignment to fulfill the following two rules:

1. Two words cannot be considered as similar if they do not share at least 50% of their characters (very short words must be considered apart).

2. Considering that words can be suppressed, added, or replaced in the *mātṛkāpāṭha*, the desired alignment has to minimise the number of those operations.

Notice that the second rule matches exactly the definition of the edit distance, but in terms of words instead of characters as is usually the case. The results provided by these two rules were approved by the philologists in charge of the Sanskrit critical edition. To illustrate our approach let us compare the following two texts: `upadi"syate mahaa .n` in the *padapāṭha* and a *mātṛkāpāṭha* with: `upadi.syata.n`. The LCS algorithm provides an alignment with an lcs of 10 that does not fulfill rule number 1.

u	p	a	d	i	"s	_	y	a	t	e	m	a	h	aa	.n
u	p	a	d	i	_	.s	y	a	t	_	_	a	_	_	.n

This involves the following conclusions:

- The string `upadi"syate` is replaced by `upadi.syat`

- The word `mahaa` is replaced by `a`

The next alignment is not optimal for the LCS criterion, because its ***lcs*** is only 9, but is preferable because the first rule is satisfied:

u	p	a	d	i	"s	_	y	a	t	e	_	m	a	h	aa	.n
u	p	a	d	i	_	.s	y	a	t	_	a	_	_	_	_	.n

- the string `upadi"syate` is replaced by `upadi.syata`
- the string `mahaa` is missing

It appears that the improvement of the initial alignment consists in asserting that the string `mahaa` is missing instead of stating that the string `maha` is replaced by `a`.

4.7. Pending problems

There are two major lacks in the software:

- If a long text is added to the *mātṛkāpāṭha* we are unable to see what are the words that compose it, because the *padapāṭha* is useless in this case.
- More important, we missed an important point at the beginning of the software conception: if a word is changed or is missing in a text, most probably *sandhi* will be changed. But the *sandhi* rules are applied at the beginning of the process, during the transformation of the *padapāṭha* into the *saṃhitapāṭha*, so we may have, in some cases, to reconsider the *sandhis* at the end of the process.

5. DISPLAYING THE RESULT

The results of the comparison program are first displayed as a `log` file as it was the best way for the necessary program tuning.

```
Paragraph 3 is Missing in File Asb2

(P3) Word 6 'paaraaya.na' is:
 - Substituted with 'paaraya.naa' in Manuscript ba2

(P3) Word 11 'saara' is:
 - Substituted with 'saadhu' in Manuscript aa

(P3) Word 17 'viv.rta' is:
 - Followed by Added word(s) 'grantha"saa' in Manuscript A3

(P3) Word 18 'guu.dha' is:
 - Missing in Manuscript A3

(P3) Word 21 'viudpanna' is:
 - Substituted with 'vyutpannaa' in Manuscript A3

(P3) Words 22 to 23 'ruupa siddhis' are:
 - Missing in Manuscript A3

(P3) Word 32 'k.rtyam' is:
 - Substituted with 'karyam' in Manuscript A3
 - Substituted with 'kaaryam' in Manuscripts aa, am4, ba2
```

After a conversion of these logged information into XML language, from which we can obtain a HTML file which can provide us an interactive version of the critical edition. Figure 3 gives an example of such a display.

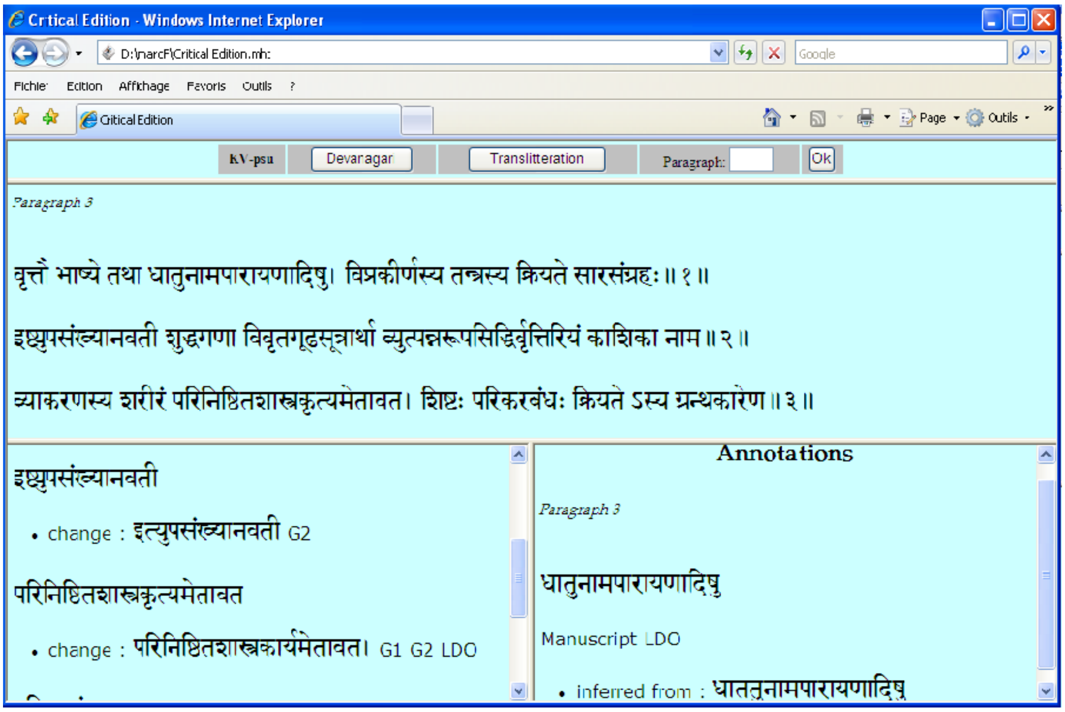


Figure 3: Example of interactive display of the results

6. CONCLUSION

In this paper we have proposed a method to compare different versions of the same Sanskrit text. The alignments provided by the LCS algorithm between two texts, considered as a sequence of characters, is not always sufficient, but provides a good initialisation for further processing that considers each of the two texts as sequences of words.

The critical edition provided by such improved alignments has been submitted to philologists and has been approved in its essential part. Nevertheless a more intense use of the software should enable us to improve and justify the setting of our empirical approach. There is also a serious need to completely rewrite the software to avoid the different dead end procedures which are still present and make the program maintenance too complicated. We also need to make more experiments for a better tuning.

The program works at a reasonable speed. With a *padapāṭha* of approximately 300 lines, and 45 different *mātṛkāpāṭha* the time needed for the comparison process is approximately 25 seconds. It seems to be quite reasonable.

However, the absence of a Sanskrit lexicon constitutes a limit to our approach: in the case of an addition of long sentences to a manuscript, it is impossible to detect words which have been added, for we can only consider the addition in terms of sequence of characters.

7. REFERENCES

CROCHEMORE, M., HANCART, C. and LECROQ, T. (2001): *Algorithmique du texte.* Vuibert, Paris.

GALE, W. A. and CHURCH, K. W. (1993): A Program for Aligning Sentences in Bilingual Corpora. *Computational Linguistics 19(3), 75–102*.

DEL VIGNA, C. and BERMENT, V. (2002) Ambiguïtés irréductibles dans les monoïdes de mots, *Actes des 9émes journées montoises d'informatique théorique*, Montpellier, Sept 2002.

CHARRAS, C. and LECROQ, T. http://www.igm.univ-mlv.fr/ lecroq/seqcomp/seqcomp.ps

HARALAMBOUS, Y. Fontes et Codages, Editions *O'reilly,* Paris 2004

HIRSCHBERG, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *CACM 18:6 341-343*.

HUET, G. (2006): *Héritage du Sanskrit: Dictionnaire Français-Sanskrit*. http://sanskrit.inria.fr/Dico.pdf.

HUET, G. (2004): Design of a Lexical Database for Sanskrit. *COLING Workshop on Electronic Dictionaries*, Geneva, 2004, pp. 8–14.

HUNT, J.W. and SZYMANSKI, T.G. (1977): A fast algorithm for computing longest common subsequence *CACM 20:5 350–353*.

INSTITUTE FOR NEW TESTAMENT TEXTUAL RESEARCH (2006): *Digital Nestle-Aland.* Münster University. http://nestlealand.uni-muenster.de/index.html.

LE POULIQUEN, M. (2007): Filiation de manuscrits Sanskrit et arbres phylogénétiques. *Submitted to Mathématiques & Sciences Humaines*.

LESK, M. E. and SCHMIDT, E., (1975): M.E. Lesk. Lex - a lexical analyzer generator. *Computing Science Technical Report 39, Bell Laboratories, Murray Hill, NJ*.

MONROY C., KAOCHUMAN R., FURUTA R, URBINBINA E., MELGOZA E., GOENKA A. (2002): Visualization of Variants in Textual Collations to Analyse the Evolution of Literary Works in the Cervantes Project. *Proceedings of the 6th European Conference, ECDL 2002. (Rome, Italy, September 2002)*. Maristella Agosti and Constantino Thanos, eds. Berlin: Springer, 2002. 638-53.

MYERS, E.W. (1986): An O(ND) Difference Algorithm and its Variations" *Algorithmica Vol. 1 No. 2, 1986, p 251.*

OHARA, R. J., and ROBINSON P.M.W. (1993): Computer-assisted methods of stemmatic analysis. *Occasional Papers of the Canterbury Tales Project, 1: 5374.* (Publication 5, Office for Humanities Communication, Oxford University.)

PAXSON, V. (1996): GNU Flex Manual, Version 2.5.3. Free Software Foundation, Cambridge, Mass. http://www.gnu.org/software/flex/manual/

RENOU L. (1996): Grammaire sanskrite: phonétique, composition, dérivation, le nom, le verbe, la phrase, Maisonneuve réimpression, Paris.

SAITOU, N. and NEI, M. (1987): The Neighbour-Joining Method: a New Method for Reconstructing Phylogenetic Trees. *Molecular Biology Evolution 4, 406–425*.

VELTHUIS, F. (1991): Devanāgarī for TEX, Version 1.2, User Manual, University of Groningen.

Index of Authors